Random Graph Modeling and Discovery

by

Neil F. Hallonquist

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

October, 2015

Abstract

In the first part of this thesis, we present a general class of models for random graphs that is applicable to a broad range of problems, including those in which graphs have complicated edge structures. These models need not be conditioned on a fixed number of vertices, as is often the case in the literature, and can be used for problems in which graphs have attributes associated with their vertices and edges. To allow structure in these models, a framework analogous to graphical models is developed for random graphs.

In the second part of this thesis, we consider the situation in which there is an unknown graph that one wants to determine. This is a common occurrence since, in general, entities in the world are not directly observable, but must be inferred from some signal. We consider a general framework for uncovering these unknown graphs by a sequence of 'tests' or 'questions'. We refer to this framework as *graph discovery*.

In the third part of this thesis, we apply graph discovery to a problem in computer vision. To evaluate how well vision systems perform, their interpretations of imagery must be compared to the true ones. Often, image interpretations can be expressed as graphs; for example, vertices can represent objects and edges can represent relationships between

ABSTRACT

objects. Thus, an image, before it is interpreted, corresponds to an unknown graph, and the interpretation of an image corresponds to graph discovery. In this work, we are interested in the evaluation of vision systems when these representation graphs are complex. We propose a visual Turing test for this purpose.

Primary Reader: Laurent Younes

Secondary Reader: Donald Geman

Acknowledgments

I would like to thank my advisor, Professor Laurent Younes, for his many years of guidance. I got my first real exposure to graphical models from his course on the topic. Even more valuable, however, were our meetings in which I had to survive his detailed inquisitions. These would usually begin with questions like, how are you defining a graph? Well, isn't it obvious how they're defined, I would think; a week later, I would realize I didn't know the answer. I would often send ideas to him in notes, and after looking at them, he would declare them ambiguous (and, on at least one occasion, blasphemous[1]). Although I would grumble about it, this feedback always made me look closer. Having an intuition about a problem was good, but never enough, and he forced me to think more carefully, to be more rigorous.

I would also like to thank Professor Donald Geman, who introduced me to many ideas. Rather early in my education, he made the mistake of saying within earshot that, if there were any new developments in the literature to know about, you probably would have already heard about them. Needless to say, the next several years were spent trying to

[1] Apparently you're not allowed to throw out all of probability theory and redefine conditional independence to to suit your needs, no matter how tempting.

convince me of the opposite. (Saying things aloud around graduate students can be dangerous.) In addition to both his philosophies and rigor, he also brought focus and direction to our projects. Whenever I would go off on (research) diversions, of which there were many, it was always made clear when I had wandered a little too far from the path. (Why are we talking about random outer measures? ...maybe you should make a to-do list.) I'm indebted to both professors for their patience.

Dedication

This thesis is dedicated to my parents, Nell and David Hallonquist.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A graph is a mathematical object that is able to encode relational information, and can be used to represent many entities in the world such as molecules, neural networks, and real-world scenes. An (undirected) graph is composed of a finite set of objects called vertices, and for each pair of vertices, specifies a binary value. If this binary value is positive, there is said to be an edge between that pair of vertices. In most applications, graphs have attributes associated with their vertices and edges; we will refer to attributed graphs simply as graphs in this thesis. (We make more formal definitions in the next chapter.) A random graph is a random variable that maps into a set of graphs.

In the first part of this thesis, we develop a random graph model, one that can specify complicated structures in graphs and in their distributions, making it applicable to a broad range of problems. This model need not be conditioned on a fixed number of vertices, as is often the case in the literature.

After developing these models, we consider the situation in which there is an unknown graph that one wants to determine. This is a common occurrence since, in general, entities in the world are not directly observable, but must be inferred from some signal. We consider a general framework for uncovering these unknown graphs by a sequence of 'tests' or 'questions'. We refer to this framework as *graph discovery*, and it composes the second part of this thesis.

Lastly, we apply graph discovery to a problem in computer vision. Computer vision is a field concerned with, among other things, the construction of vision systems meant to comprehend imagery; comprehension of an image refers to the ability to correctly produce a compressed representation of it. This representation, in many cases, can be formulated as an (attributed) graph. For example, vertices can represent objects (e.g., people and vehicles), and edges can represent relationships between objects (e.g., holding hands, driving, etc.). Thus, an image, before it is interpreted, corresponds to an unknown graph, and the interpretation of an image corresponds to graph discovery. In this work, we are interested in the evaluation of vision systems when these representation graphs are complex. We propose a visual Turing test for this purpose, and this composes the third part of this thesis.

1.1 Random Graphs

There is a large literature about random graphs. In this section, we give a brief overview of models in the literature, and discuss some of their shortcomings, motivating our work.

1.1.1 Background

The most commonly studied random graph model is the Erdős-Rényi model ([Erdős & Rényi, 1959], [Gilbert, 1959]). This is a model for conditional distributions in which, for a given set of vertices, a distribution is placed over the possible edges. It makes the invariance assumption that, for any two vertices, the probability of an edge between them is independent of the other edges in the graph, and further, this probability is the same for all edges. This classic model, due to its simplicity, is conducive to mathematical analysis; its asymptotic behavior (i.e, its behavior as the number of vertices becomes large) has been researched extensively.

There are many ways in which the Erdős-Rényi model can be extended. One such extension is the *stochastic blockmodel* [Holland et al., 1983]. This model is for conditional distributions over the edges, given vertices, where each vertex has a label (e.g., a color) associated with it. Similar to the Erdős-Rényi model, for any two vertices, the probability of an edge between them is independent of the other edges in the graph; unlike the the Erdős-Rényi model, this probability depends on the labels of those two vertices.

An extension of the stochastic blockmodel is the *mixed membership stochastic blockmodel* [Airoldi et al., 2009]. In this model, instead of associating each vertex with a fixed label, each vertex is associated with a probability vector over the possible labels. Given a set of vertices (and their label probability vectors), a set of edges can be sampled as follows: for each pair of vertices, first sample their respective labels, then sample from a Bernoulli distribution that depends on these labels. Thus, loosely speaking, this model adds addi-

tional randomness to the distributions. Another extension of the stochastic blockmodel is the *latent space model* [Hoff at al., 2002], where instead of associating vertices with labels from a finite set, they are instead associated with positions in a Euclidean space; given the position of two vertices, the probability of an edge between them only depends on their distance.

A general class of random graph models, of which the above models fall within, is the exponential family ([Holland & Leinhardt, 1981], [Robins, 2011], [Snijders et al., 2006]). A well-known example is the Frank and Strauss model [Frank & Strauss, 1986], also a model for conditional distributions, specifying the probability of having some set of edges, given vertices. Since the randomness is only over the edges, a graphical model can be applied in which there is a random variable for each pair of vertices, specifying the presence or absence of an edge. These random variables are conditionally independent, in this model, if they do not share a common vertex.

1.1.2 Other Literature

In this section, we review models from outside the mainstream random graph community that were designed for graphs with complicated attributes. One of the first such models was developed by Ulf Grenander under the name *pattern theory* ([Grenander, 1976], [Grenander, 1997], [Grenander & Miller, 2007]). This work was motivated by the desire to formalize the concept of a pattern within a mathematical framework. A large collection of natural and man-made patterns is shown in [Grenander, 1996]. Examples range from tex-

tures to leaf shapes to human language. In each of these examples, every particular instance of the given pattern can be represented by a graph. These instances have natural variations, and so the mathematical framework for describing these patterns is probabilistic, i.e. a random graph model. The model developed was based on applying Markov random fields to graphs. Learning and inference are often difficult in this model, limiting its practical use.

Later, random graph models were developed within the field of *relational statistical learning*. In particular, techniques such as Probabilistic Relational Models [Friendman et al., 1999], Relational Markov Networks [Taskar et al., 2002], and Probabilistic Entity-Relationship Models [Heckerman et al., 2007], were specifically designed for modeling entities that are representable as graphs. These models specify conditional distributions, applying graphical models in which: (1) for each vertex, there is a random variable representing its attributes; and (2) for each pair of vertices, there is a random variable representing their edge attributes. (This is an approach similar to the one taken in the Frank and Strauss model).

1.1.3 Issues

Suppose we want to learn a distribution over some graph space. This distribution cannot be directly modeled with graphical models because these were designed for multivariate random variables (with a fixed number of components). To avoid this issue, most random graph models in the literature transform the problem into one in which graphical models can be applied. This is done by only modeling a selected set of conditional distributions, for

example, the set of distributions in which each is conditioned on some number of vertices. Aside from the fact that many applications simply require full distributions, problems with this approach include: (1) there are complicated consistency issues; a distribution may not exist that could produce a given set of conditional distributions; and (2) this partial modeling, loosely speaking, cannot capture important structures in distributions (e.g., there may be invariances within a full distribution that are difficult to encode within conditional distributions). To correct these issues, graphical models cannot be used for this problem; we need statistical models specifically designed for general graph spaces. Suppose we have a graph space $\mathcal{G}$ in which graphs may differ in their order (i.e., graphs in this space may vary in their number of vertices); in this thesis, we want to develop distributions over this type of space.

In addition, we want models that are applicable to problems in which: (a) graphs have complex edge structures; and (b) graphs have attributes associated to their vertices and edges. To handle these problems, expressive models are necessary (i.e., models containing a large set of distributions). To make learning feasible in these models, it becomes imperative to specify structure in them as well.

1.1.4 Factorization

To specify structure in random graph distributions, we look to the standard methods used in multivariate random variables for insight. Suppose we have a random variable $\mathbf{X}$ taking values in $\mathcal{X} = \{0, 1\}^n$. In general, its distribution has $2^n - 1$ parameters that need

to be specified. If the value of n is not small, learning this number of parameters, in most real-world problems, is infeasible; hence, the need to control complexity. This has led to the wide-spread use of graphical models, a framework that uses factorization to simplify distributions. In this framework, joint distributions are specified by simpler functions, and more specifically, the probability of any $X \in \mathcal{X}$ is uniquely determined by a product of functions over subsets of its components.

Now, suppose we have a random graph $\mathbf{G}$ taking values in some finite graph space $\mathcal{G}$. In general, its distribution has $|\mathcal{G}| - 1$ parameters that need to be specified, and again, clearly there is a need to control complexity. Similar to graphical models, we can simplify distributions through the use of factorization: the probability of any graph $G \in \mathcal{G}$ can be uniquely determined by a product of simpler functions over its subgraphs. Thus, we can create a general framework for random graphs analogous to that of graphical models. Indeed, just as graphs can be used to represent the factorization in graphical models, graphs can also be used to represent the factorizations in random graphs. We study these models in chapter 2.

1.2 Graph Discovery

Suppose we have an *unknown* graph that we want to determine by asking questions: the answer to any question, when asked about this graph, provides information about it. We refer to *graph discovery* as the problem of uncovering unknown graphs by sequential

questioning. Applications include problems in which graphs must be efficiently determined from some signal.

Let a question be a function of the form $f : \mathcal{G} \to \{0, 1\}$, taking graphs from a graph space $\mathcal{G}$ to their binary answer. For questions to be efficient, we must have some knowledge of the likelihoods of graphs; assume we have a distribution over $\mathcal{G}$. Using this, we can make sequences of probing and incisive questions for an unknown graph.

1.2.1 Instantiation

Instantiation of a vertex refers to: (1) the discovery of some set of attributes that distinguish that vertex from all others in the unknown graph; and (2) the denotation of this unique vertex with a label (e.g. vertex u). A sequence of questions can be chosen with the goal of instantiating vertices. Once a vertex is instantiated, it may be further explored with more specialized questions.

1.2.2 Predictability

Let $\mathcal{F}$ be some space of binary questions. We refer to a function that sequentially selects questions from $\mathcal{F}$ as a *query-engine*. A natural criteria for selecting questions is based on maximizing the expected information about the unknown graph, and this corresponds to the selection of questions that are *unpredictable*. Roughly speaking, a question is unpredictable if its answer cannot be reliably guessed, even given the answers to all previously asked

questions. Details of graph discovery are given in chapter 4.

1.3 A Visual Turing Test

Let a vision system be a machine that takes an image and produces a compressed representation of it. To evaluate how well vision systems perform this task, the representations they produce must be compared to the true ones. In general, image representations can be expressed as graphs; for example, vertices can represent objects and edges can represent relationships between objects. Thus, an image, before it is interpreted, corresponds to an unknown graph. The problem we confront is that, as the representation space is made increasingly complex (e.g. having relationships between objects) and the image space increasingly unconstrained (e.g. involving real-world scenes), it becomes increasingly costly for human annotators to record the true representations. Thus, in any practical implementation, evaluations can only be over partially determined[1] representations of the imagery. The question, loosely speaking, is how to select the parts to be tested.

In the proposed visual Turing test, evaluations are based on binary questions about the representations[2]. For each question, a human annotator provides the true answer and the vision system provides its best guess. Since efficiency is paramount in any evaluation, these binary questions cannot be arbitrary; not all questions provide the same amount of

[1] This situation is analogous to that of a teacher evaluating students in a course; an exam cannot cover every detail because the amount of material is too great, and so the teacher must test over a selected subset.

[2] We will define questions to be functions of the representation graph. Equivalently, however, questions can be defined as functions of the image (since the representation graph is also a function of the image).

information. For example, some questions can be answered correctly most of the time without even looking at the image (e.g., questions designed to discern fine details usually have a negative answer). We make the following assumption: the questions that provide the most information about the vision system under evaluation are also the same questions that provide the most information about the representation graph. Although not entirely valid[3], this assumption provides a useful heuristic. Under it, an evaluation can be formulated as a graph discovery problem; we select questions that maximize the expected information about the unknown graph. Equivalently, we select questions that are unpredictable. These ideas are discussed in chapter 5.

1.3.1 Models

The main difficulty in implementing the proposed visual Turing test is in estimating unpredictable questions. To estimate these, we need to estimate the conditional probability of questions having a positive answer given the previous history of questions and answers. As the number of questions in the history increases, this estimation becomes increasingly difficult. For example, empirical estimates become infeasible because the number of graphs in any (finite) dataset that cohere with the history rapidly decreases as the length of the history increases. To estimate these conditional probabilities, we must use invariance assumptions; loosely, these serve to increase the amount of information that can be extracted

[3]For example, this assumption means that vision systems are not tested for anomaly detection, i.e. unlikely events. Questions about them, in general, have a low expected information about the representation graph. Nonetheless, these might be questions that provide a lot of information about the vision system.

from a dataset for estimates.

However, care must be taken when using invariance assumptions as they can cause the estimated conditional probabilities to be inconsistent. A set of conditional probabilities is inconsistent if there does not exist a probability distribution that could have produced it. Invariance assumptions can result in inconsistent estimates if they are applied to each estimate individually, not accounting for their effect on the set of estimates as a whole. To ensure consistency, we incorporate the invariance assumptions into a statistical model. In particular, we estimate a random graph model for this purpose. The conditional probabilities are then inferred from this model, and used to produce unpredictable question sequences. The inference is performed by conditionally sampling from the model (i.e., sampling from its conditional distributions). In general, this sampling is difficult because the evidence resulting from question sequences is complex. However, under certain assumptions on the question space, the conditional sampling becomes amenable to computation. This work is discussed in chapter 6.

1.4 Dissertation Contributions

The contributions in this thesis are:

1. We develop a general random graph model applicable to a wide range of problems.

2. We introduce a framework for uncovering unknown graphs.

3. We develop a novel test for evaluating vision systems based on binary questions.

4. To our knowledge, we are the first to model real-world scenes using random graphs.

5. We develop a conditional sampling algorithm for random graphs, where conditioning is based on sequences of questions and answers. We assume certain restrictions on the questions that can be asked.

Chapter 2

Random Graphs

In this chapter, we present a general class of models for random graphs. We first define, for graph distributions, a Gibbs form based on the projection of graphs to their subgraphs; then, for modeling purposes, we consider methods for specifying structure. These models can be used for creating complex distributions (e.g., distributions that place significant mass on graphs with complicated edge structures). These models are illustrated with several examples. Sampling and learning algorithms are provided in the next chapter.

2.1 Graphs

Suppose we have a vertex space Λ_V and an edge space Λ_E. We define a graph to be a couple of the form $G = (V, E)$, where V is a set of vertices and E is a function assigning

an edge value to every pair of vertices:

$$V \subset \Lambda_V$$

$$E : V \times V \to \Lambda_E.$$

Hence, every vertex is unique, i.e. no two vertices can share the same value in Λ_V. We assume the edge space Λ_E contains a distinguished element that represents the 'absence' of an edge (e.g. the value 0). If a graph has no vertices, i.e., $|V| = 0$, we will denote it by $\emptyset$ and refer to it as the empty graph. For simplicity, we assume there are no self loops. That is, there are no edges between a vertex and itself (i.e., $E(v, v) = 0$ for all $v \in V$).

Example 2.1: Let the vertex space be $\Lambda_V = \mathcal{C} \times \mathcal{L}$, where:

$$\mathcal{C} = \{c_1, c_2\}$$

$$\mathcal{L} = \{1, \ldots, p\} \times \{1, \ldots, p\}$$

and each c_i represents a color, and $\mathcal{L}$ represents a location space, a two dimensional grid of size p. Let the edge space be $\Lambda_E = \{0, 1, 2\}$, where the value 0 represents the absence of an edge. See figure 2.1 for an example graph.

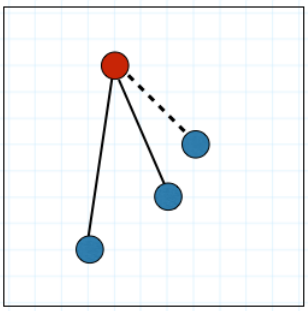


Figure 2.1: An example graph that uses the vertex and edge space described in example 2.1. The edge values 1 and 2 are represented by lines and dotted lines, respectively. The vertices are colored either red or blue, and take locations on a grid.

In most real-world applications, graphs have attributes associated with their vertices and edges; in this case, the vertex space Λ_V and edge space Λ_E can be complicated. Furthermore, for modeling purposes, sometimes it is important to include latent variables into these spaces. We consider more examples later in this chapter.

2.2 A Set of Sufficient Statistics

We now consider a very general, but important set of sufficient statistics of a graph; these will be used in our statistical models. Let the subgraph of $G = (V, E)$ *induced* by a subset $V' \subseteq V$ of its vertices be the graph $G' = (V', E')$, where $E' = E|_{V' \times V'}$ is the restriction of E. We let $G' = G(V')$ denote the subgraph of G induced by $V' \subseteq V$.

For a graph $G = (V, E)$, define the set of induced subgraphs of order 1 as:

$$S_1(G) = \{G' = G(V') \ : \ V' \subseteq V \text{ and } |V'| = 1\}.$$

Each element of this set is a graph containing a single vertex. Thus, this set is simply the set of vertices of the graph G. Define the set of induced subgraphs of order 2 as:

$$S_2(G) = \{G' = G(V') \ : \ V' \subseteq V \text{ and } |V'| = 2\}.$$

This is the set of all subgraphs of G with two vertices. In general, define the set of subgraphs of order k as:

$$S_k(G) = \{G' = G(V') \ : \ V' \subseteq V \text{ and } |V'| = k\}.$$

We now use these statistics to define Gibbs distribution for graphs.

2.3 Gibbs Distribution on Graphs

Let $\mathcal{G}$ be a countably infinite space of graphs. A Gibbs distribution for the graph space $\mathcal{G}$ is the following:

Definition (Gibbs Distribution): a probability mass function (pmf) P over a graph space $\mathcal{G}$ is a *Gibbs distribution* if it can be written in the form

$$P(G) = \exp\left[\psi_0 + \sum_{G' \in S_1(G)} \psi_1(G') + \sum_{G' \in S_2(G)} \psi_2(G') + \dots\right] \tag{2.1}$$

where $\psi_k : \mathcal{G}^{(k)} \to \mathbb{R} \cup \{-\infty\}$ is called the potential of order k, and $\mathcal{G}^{(k)}$ denotes the space of graphs of order k, i.e. $\mathcal{G}^{(k)} = \{(V, E) \in \mathcal{G} \ : \ |V| = k\}$.

We give some examples in which classical models are expressed in this form.

Example 2.2 (The Erdős-Rényi model) [Erdős & Rényi, 1959], [Gilbert, 1959]: Let $\mathcal{G}$ be a standard graph space (i.e. the vertex space $\Lambda_V = \mathbb{N}$ is the set of natural numbers and the edge space $\Lambda_E = \{0, 1\}$.) The Erdős-Rényi model is a conditional distribution specifying the probability of edges E given a set of vertices V. It makes the invariance assumption that, for any two vertices, the probability of an edge between them is independent of the other edges in the graph:

$$P(E|V) = \exp\left[\sum_{G' \in S_2((V,E))} \psi_2(G')\right]$$

where

$$\psi_2(G') = \begin{cases} \log(p), & \text{if } G' \text{ has an edge} \\ \log(1-p), & \text{otherwise} \end{cases}$$

and $p \in [0, 1]$.

Example 2.3 (The stochastic blockmodel) [Holland et al., 1983]: Let $\mathcal{G}$ be a graph space

where the vertex space is $\Lambda_V = \{1, \ldots, l\} \times \mathbb{N}$, where the first component corresponds to some label, and the edge space is $\Lambda_E = \{0, 1\}$. The stochastic blockmodel is also a conditional distribution specifying the probability of edges E given a set of vertices V. It makes the invariance assumption that, for any two vertices, the probability of an edge between them depends on only the label of those two vertices:

$$P(E|V) = \exp\left[\sum_{G' \in S_2((V,E))} \psi_2(G')\right]$$

where

$$\psi_2(G') = \begin{cases} \log\ p(a_1, a_2), & \text{if } G' \text{ has an edge} \\ \log(1 - p(a_1, a_2)), & \text{otherwise} \end{cases}$$

where a_1, a_2 are the labels of the two vertices in $G' \in \mathcal{G}^{(2)}$ and $p : \{1, \ldots, l\}^2 \to [0, 1]$ is a symmetric function.

2.4 The Positivity Condition

In this section, we define a positivity condition for distributions. For a graph G, let $S(G)$ denote the set of all subgraphs of G that can be induced by subsets of its vertices.

That is, let

$$S(G) = S_0(G) \cup S_1(G) \cup \ldots,$$

where $S_0(G) = \{\emptyset\}$. Before defining positivity, we must make sure the graph space is convenient to work with:

Definition (Projectable Space): A graph space $\mathcal{G}$ is *projectable* if:

$$G \in \mathcal{G} \implies G' \in \mathcal{G} \text{ for all } G' \in S(G).$$

Now we define positivity of a distribution:

Definition (Positivity Condition): Let P be a real function over a projectable graph space $\mathcal{G}$. The function P is said to satisfy the *positivity condition* if, for all $G \in \mathcal{G}$, we have:

$$P(G) > 0 \implies P(G') > 0 \text{ for all } G' \in S(G).$$

Henceforth, we assume that every graph space $\mathcal{G}$ is projectable.

2.5 Compact Distributions

As mentioned in the previous chapter, it is imperative for distributions to have compact representations. For a graph space $\mathcal{G}$, a distribution over it can always be specified by directly assigning a probability to each graph $G \in \mathcal{G}$. Of course, this is impractical for all but the smallest graph spaces, and hence we must design models in which complexity can be controlled. This can be accomplished by assuming distributions have some form of factorization.

Consider multivariate random variables: suppose we have a random variable $\mathbf{X}$ taking values in $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, where each $\mathcal{X}_i$ is finite. To define a distribution over $\mathcal{X}$, we will assume it equals some product of simpler functions (i.e., functions that have smaller domains than $\mathcal{X}$). To define these simpler functions, we use projections of the form $\pi_C : \mathcal{X} \to \mathcal{X}_C$, where $C \subseteq \{1, \ldots, n\}$ and $\mathcal{X}_C = \prod_{i \in C} \mathcal{X}_i$, and take elements in $\mathcal{X}$ to their components. Using these projections, we can define factors of the form $f_C : \mathcal{X}_C \to \mathbb{R}^+$, and a distribution P factorizes over $\mathcal{C} \subseteq \mathbb{P}(\{1, \ldots, n\})$ if it can be written as:

$$\begin{aligned} P(X) &= \frac{1}{Z} \prod_{C \in \mathcal{C}} f_C(\pi_C(X)) \\ &= \frac{1}{Z} \prod_{C \in \mathcal{C}} f_C(X_C), \end{aligned}$$

where $X_C \equiv \pi_C(X)$. Structure can be specified in this model by the choice of factors. For a given model, complexity can be reduced through the removal of factors (i.e., removing elements from the set $\mathcal{C}$).

Now suppose we have a random graph $\mathbf{G}$ taking values in $\mathcal{G}$. To define a notion of factorization for a distribution over a graph space, we need a set of projections:

Definition (Projection A): Let $V \subseteq \Lambda_V$ be a set of vertices. Define the projection $\pi_V : \mathcal{G} \to \mathcal{G}_V$, where $\mathcal{G}_V = \{G \in \mathcal{G} \mid V(G) = V\} \cup \{\emptyset\}$, as:

$$\pi_V(G) = \begin{cases} G(V), & \text{if } V \subseteq V(G) \\ \emptyset, & \text{if } V \not\subseteq V(G). \end{cases}$$

In other words, the projection π_V maps a graph G to its subgraph induced by V if these vertices are in the graph; otherwise, G is mapped to the empty graph. With these projections, a distribution can then be defined as a product of factors of the form $f_V : \mathcal{G}_V \to \mathbb{R}^+$; a distribution P factorizes over $\mathcal{V} \subseteq \mathbb{P}(\Lambda_V)$ if it can be written as:

$$\begin{aligned} P(G) &= \frac{1}{Z} \prod_{V \in \mathcal{V}} f_V(\pi_V(G)) \\ &= \frac{1}{Z} \prod_{\substack{V \in \mathcal{V} \\ V \subseteq V(G)}} f_V(G_V), \end{aligned}$$

where $G_V \equiv \pi_V(G)$, and where we are assuming $f_V(\emptyset) = 1$ for all V.

As with graphical models, structure can be specified through the choice of factors. For random graph modeling, however, we find it more natural to allow slightly more structure and allow individual graphs to be assigned a factor value of one (i.e., allow one to set

$f_V(G) = 1$ for some $G \in \mathcal{G}_V$). Equivalently, we can assign all graphs to have a factor value of one except for those in some subset $\mathcal{G}_{\text{basis}} \subset \mathcal{G}$. Thus, we can control complexity through the size of the basis $\mathcal{G}_{\text{basis}}$, and returning to exponential form, distributions can be expressed as:

$$P(G) = \frac{1}{Z} \exp\left[\sum_{G' \in \mathcal{C}(G)} \phi(G')\right], \tag{2.2}$$

where $\phi : \mathcal{G}_{\text{basis}} \to \mathbb{R} \cup \{-\infty\}$ and $\mathcal{C}(G) \equiv S(G) \cap \mathcal{G}_{\text{basis}}$.

Finally, we mention there are other projections that could be used here; in section 2.8, we discuss an alternative projection that has some desirable properties (e.g., can be used to define marginal random variables for graphs).

2.5.1 Additional Structure

The model given in (2.2) can be further simplified by assuming the function ϕ has some structure. This can be done in many ways; the simplest is to assign the same function value to graphs that are similar in some sense. For example, we might want graphs that are isomorphic to each other to have equal values (i.e., setting $\phi(G_1) = \phi(G_2)$ for all $G_1, G_2 \in \mathcal{G}_{\text{basis}}$ that are isomorphic). More generally, we can specify structure in ϕ by assuming an additive relationship of the form:

$$\phi(G) = \sum_{k=1}^{K} \lambda_k I_{\{G \in \mathcal{D}_k\}},$$

where each $\mathcal{D}_k \subset \mathcal{G}_{\text{basis}}$ is a subset of the basis and each λ_k a real number. Then the model in equation (2.2) simplifies to:

$$P(G) = \frac{1}{Z} \exp\left[\sum_{k=1}^{K} \lambda_k U_k(G)\right], \tag{2.3}$$

where $U_k(G) = \#\{G' \in S(G) \ : \ G' \in \mathcal{D}_k\}$ is the number of subgraphs of type k in the graph G. We will find it convenient to reformulate each set $\mathcal{D}_k$ as a binary function: define a function $R_k : \mathcal{G} \to \{0, 1\}$ such that

$$R_k(G) = 1 \iff G \in \mathcal{D}_k.$$

Then, equivalently, we have that $U_k(G) = \#\{G' \in S(G) \ : \ R_k(G') = 1\}$. We refer to the binary functions R_k as *compatibility maps*. We now consider methods for specifying these maps.

2.5.2 Graph Isomorphisms

One way to define compatibility maps is based on graph isomorphisms. Two graphs are said to be isomorphic if they share the same edge structure:

Definition 1: (Graph Isomorphism) A graph $G = (V, E)$ is *isomorphic* to a graph $G' =$

(V', E') if there exists a bijection $f : V \to V'$ such that:

$$E(v, v') = E'(f(v), f(v')) \text{ for all } v, v' \in V.$$

Two graphs that are isomorphic are denoted by $G \simeq G'$.

This definition compares two graphs with each other by comparing their parts, and more specifically, compares their second-order subgraphs. Given a set of template graphs, we can define compatibility maps based on the set of graphs that are isomorphic to them. That is, given a set of template graphs $\{T_1, \ldots, T_K\}$, where each $T_k \in \mathcal{G}_{\text{basis}}$, we may define each map R_k as follows:

$$R_k(G) = 1 \iff G \simeq T_k.$$

2.5.3 Relations Induced by Parts

For modeling purposes, often comparing graphs based on the above graph isomorphism is too rigid. In this section, we consider more general methods for comparing graphs based on parts. In general, this approach is valuable because specifying how parts compare is easier than specifying how whole graphs compare. The simplest possible method is based on vertices:

Definition 2(a): (First-order Isomorphism) Let $\sim_1$ be a binary relation on the vertex space Λ_V. A graph G is *isomorphic* to a graph G_0 with respect to $\sim_1$ if there exists a bijection $f : V(G_0) \to V(G)$ such that:

$$v \sim_1 f(v) \text{ for all } v \in V(G_0).$$

Example 2.4: Suppose we have a vertex space is $\Lambda_V = \mathcal{C} \times \mathcal{L}$, where $\mathcal{C}$ is some set of colors and $\mathcal{L}$ is some set of locations. Define the following binary relation:

$$v \sim_1 v' \iff \text{color}(v) = \text{color}(v'),$$

where color(v) denotes the projection of the vertex v to its color component. Hence, two graphs are isomorphic with respect to $\sim_1$ if they have the same composition of colors in their vertices.

The next simplest comparison is based on second-order subgraphs:

Definition 2(b): (Second-order Isomorphism) Let $\sim_1$ and $\sim_2$ be binary relations on $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$, respectively. A graph G is *isomorphic* to a graph G_0 with respect to $(\sim_1, \sim_2)$ if there exists a bijection $f : V(G_0) \to V(G)$ such that:

1. $v \sim_1 f(v)$ for all $v \in V(G_0)$.

2. $G_0(\{v, v'\}) \sim_2 G(\{f(v), f(v')\})$ for all $v, v' \in V(G_0)$

We may extend these definitions to higher-order subgraphs in a straightforward way. To summarize, we presented some isomorphisms that can be used in specifying when graphs are similar to each other. We will make use of them to specify compatibility maps in the examples presented in the next section. To be more specific, we will specify a set of template graphs $\{T_1, \ldots, T_K\}$, where each $T_k \in \mathcal{G}_{\text{basis}}$, and a set of relations $(\sim_1, \sim_2)$; then each map R_k can be defined as follows:

$$R_k(G) = 1 \iff G \text{ and } T_k \text{ are isomorphic w.r.t. } (\sim_1, \sim_2).$$

Finally, we mention that these definitions can be adjusted so as not to assume symmetry.

2.6 Master Interaction Function

In defining distributions over a graph spaces, often it will be useful to reduce the size of the graph space, removing graphs that have zero probability. One way to do this, assuming that the edge space Λ_E has a partial ordering, is to define a function that restricts the edge configurations allowed in graphs:

Definition: (Master Interaction Functions) A *master interactions function* is a function of

the form $U : \Lambda_V \times \Lambda_V \to \Lambda_E$. A graph $G = (V, E)$ is said to *respect* a master interactions function U if, for all $v, v' \in V$, we have $E(v, v') \leq U(v, v')$.

We use master interactions functions to restrict graph spaces to only those graphs that respect it. That is, we will define graph spaces of the form:

$$\mathcal{G} = \left\{ (V, E) \ : \begin{array}{l} V \subset \Lambda_V, \ |V| \leq N \\ E : V \times V \to \Lambda_E \\ E(v, v') \leq U(v, v') \text{ for all } v, v' \in V \end{array} \right\}.$$

Example 2.5: Suppose the vertex space $\Lambda_V = \mathbb{N}$ and the edge space $\Lambda_E = \{0, 1\}$. Define a master interactions function U as follows:

$$U(v, v') = 0 \iff |v - v'| > t,$$

where $t \in \mathbb{R}^+$. This master interactions function ensures there is no edge between vertices that are farther apart than t.

2.7 Examples

In this section, we illustrate the above ideas with some examples. In each example, we specify a vertex space Λ_V and edge space Λ_E, and then, to define a distribution, we specify a set of templates $\{T_1, \ldots, T_K\}$ and their compatibility maps, and the model takes the form of equation (2.3). The sampling and learning algorithms are discussed in chapter 3.

2.7.1 Example 1: Grid Graphs

We consider grid-like graphs such as the one shown in figure 2.2. Let the vertex space be $\Lambda_V = \mathbb{N} \times \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers, and let $\Lambda_E = \{0, 1\}$, specifying the absence of an edge or the presence of an edge, respectively. We can specify the master interactions function U to take pairs of vertices that cannot have an edge to the value 0, and pairs that can have an edge to the value 1. Define U as follows:

$$U(v, v') = 1 \iff |v_1 - v'_1| \leq 1 \text{ and } |v_2 - v'_2| \leq 1,$$

where $v = (v_1, v_2) \in \Lambda_V$ and $v' = (v'_1, v'_2) \in \Lambda_V$. Hence, this master interactions function U ensures the graph space $\mathcal{G}$ only contains grid-like graphs.

A possible set of templates is shown in table 2.1. Each template T_k in these tables specifies a compatibility map based on graphs that are isomorphic to T_k. Here, we made the following design choices. First, we have limited the order of the template graphs to fourth order and lower (i.e. graphs such that $|G| \leq 4$). Secondly, to make computation

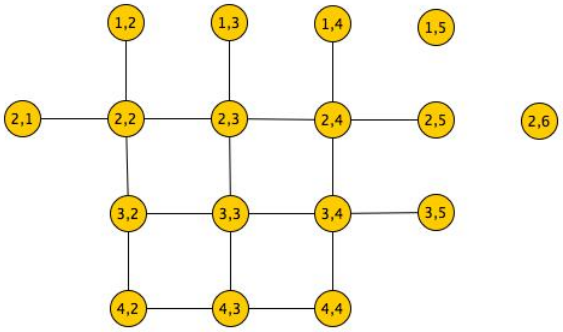


Figure 2.2: An example of a grid-like graph.

feasible, we apply a 'locality' principle in which only connected graphs are used as templates. Since unconnected graphs constitute the vast majority of the subgraphs in $S(G)$ for any given graph $G \in \mathcal{G}$, the restriction to only these is necessary for computational reasons. For example, consider the second-order subgraphs in the graph in figure 2.2; there are $|S_2(G)| = \binom{17}{2} = 136$ subgraphs of this order, but only 18 of them are connected. If we consider higher-order subgraphs, this gap widens.

Given these templates, the number of subgraphs that correspond to a given pattern can be calculated for any graph $G \in \mathcal{G}$, and hence its probability can be calculated. For example, for the graph G in figure 2.2, the probability is expressed as follows:

$$\begin{aligned} P(G) &= \frac{1}{Z} \exp\left[\sum_{k=1}^{K} \lambda_k U_k(G)\right] \\ &= \frac{1}{Z} \exp\left[17\lambda_1 + 18\lambda_2 + 26\lambda_3 + 4\lambda_4 + 45\lambda_5 + 20\lambda_6\right]. \end{aligned}$$

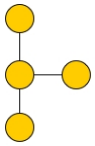
	λ_1
	λ_2
	λ_3
	λ_4
	λ_5
	λ_6

Table 2.1: The set of connected graphs that are used as templates; the compatibility maps are based on graphs that are isomorphic to these templates.

2.7.2 Example 2: 'Molecule' Graphs

We consider an example in which the graph space $\mathcal{G}$ is composed of graphs that loosely resemble molecules in appearance. An example is shown in figure 2.3.

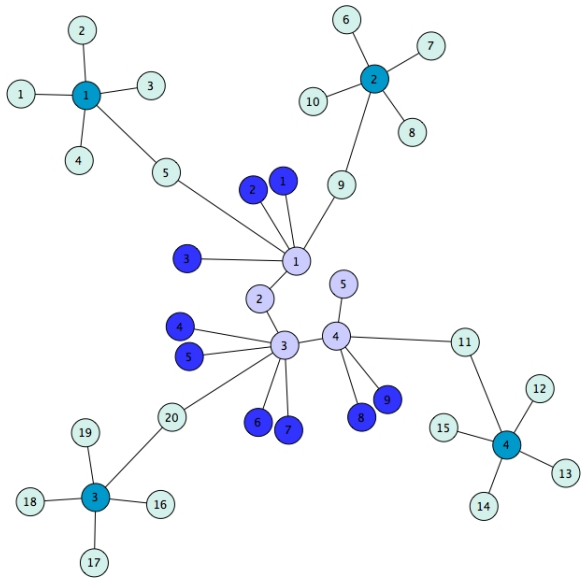


Figure 2.3: An example of a 'molecule' graph. This is an artificial graph, made only for illustration.

In this example, a value in $C = \{c_1, c_2, c_3, c_4\}$, where each c_i represents a color, and an index in $\mathbb{N}$ is associated with each vertex. The vertex space is $\Lambda_V = C \times \mathbb{N}$ and the edge space is $\Lambda_E = \{0, 1\}$, specifying the absence or presence of an edge, respectively.

We can specify the master interactions function U to take pairs of vertices that cannot have an edge to the value 0 and take pairs of vertices that can have an edge to the value 1. For example, we might want to specify that vertices with the same color cannot have an edge between them for certain colors (e.g. set $U(v, v') = 0$ for all $v = (c_1, n_1)$ and $v' = (c_1, n_2)$, $n_1, n_2 \in \mathbb{N}$). Similarly, we might want to specify that vertices of particular

different colors can have an edge between them (e.g. set $U(v, v') = 1$ for all $v = (c_1, n_1)$ and $v' = (c_2, n_2)$, $n_1, n_2 \in \mathbb{N}$).

Let T_k be some template graph. We will define a compatibility map $R_k : \mathcal{G} \to \{0, 1\}$ based on two relations $\{\sim_1, \sim_2\}$. These relations will ensure color consistency and spatial invariance respectively. Define them as follows:

1. **Color**: We want to consider a vertex to be similar to another vertex only if their colors match. Define a relation $\sim_1$ on $\mathcal{G}^{(1)}$ as follows:

$$G \sim_1 G' \iff \text{color}(v) = \text{color}(v')$$

   where color(v) is the projection of v to its first factor, $G, G' \in \mathcal{G}^{(1)}$, and v and v' is each graphs vertex, respectively.

2. **Structure Invariance**: We want to consider a graph to be compatible with a template if the edge structure is the same. Define a relation $\sim_2$ on $\mathcal{G}^{(2)}$ as follows:

$$G \sim_2 G' \iff E(v_0, v_1) = E'(v'_0, v'_1)$$

   where $G, G' \in \mathcal{G}^{(2)}$, and where $V(G) = \{v_0, v_1\}$ and $V(G') = \{v'_0, v'_1\}$ is each graphs vertices, respectively.

A possible set of templates and their corresponding parameters is shown in table 2.2. Given these templates, the number of subgraphs that correspond to a given pattern can be

calculated for any graph $G \in \mathcal{G}$, and hence its (unnormalized) probability can be calculated. For the graph G in figure 2.3, the probability can be expressed as follows:

$$\begin{aligned} P(G) &= \frac{1}{Z} \exp\left[\sum_{k=1}^{K} \lambda_k U_k(G)\right] \\ &= \frac{1}{Z} \exp\left[20\lambda_1 + 4\lambda_2 + 9\lambda_3 + 5\lambda_4 + 4\lambda_5 + 20\lambda_6 + 4\lambda_7 + 9\lambda_8 + 6\lambda_9 + 4\lambda_{10}\right]. \end{aligned}$$

Notice that in this example, the attributes (i.e., the colors associated with vertices) allow distributions in which, loosely speaking, typical samples have complex structure even despite the fact that the basis does not contain high-order graphs. For example, the edge structure in these graphs are very unlikely to have been generated by independent coin flips as in an Erdős-Rényi model. If the vertices did not have these attributes and we wanted to define a distribution that has equivalent probabilities as in this example, (e.g. assign the same probability to the unattributed version[1] of the graph in figure 2.3), it would be necessary for any basis to contain graphs of much higher orders than those in the basis used in this example. Hence, we see that attributes are important latent variables even if one only wants to define distributions over unattributed graph spaces. Thus, ideas contained in latent position models ([Hoff et al., 2002]) and latent stochastic blockmodels ([Airoldi et al., 2009], [Latouche et al., 2011]) can be incorporated within the framework here.

[1] That is, a graph with the same edge configuration.

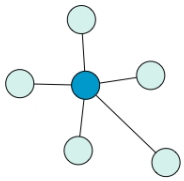
	λ_1
	λ_2
	λ_3
	λ_4
	λ_5
	λ_6
	λ_7
	λ_8
	λ_9
	λ_{10}

Table 2.2: The set of graphs that are used as templates. These are used to specify the compatibility maps based on graphs that are isomorphic to them with respect to the binary relations described in example 2.

2.7.3 Example 3: Mouse Visual Cortex

A graph G_0 that corresponds to the visual cortex of a mouse and is shown in figure 2.4.

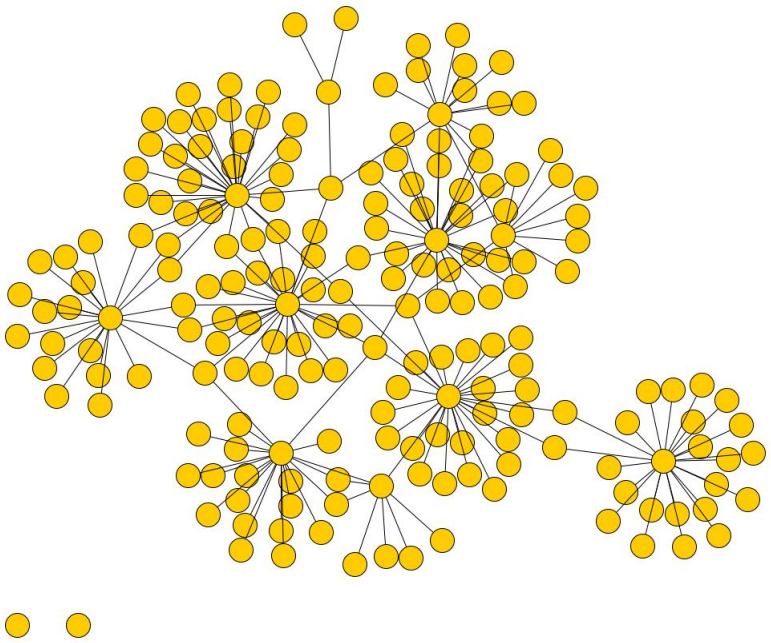


Figure 2.4: A mouse visual cortex, which we label as G_0. (Source [Open Connectome Project]).

In order to model mouse visual cortexes, we consider a hierarchical model; we begin by modeling parts (i.e. interesting subgraphs) that compose these visual cortexes. From the graph G_0, we extract subgraphs of G_0 that exist in the following graph space:

$$\mathcal{G} = \left\{ G = (V, E) \ : \begin{array}{l} |V| = 25 \\ G \text{ is connected} \\ \text{diameter}(G) \leq 5 \end{array} \right\}.$$

Some examples are shown in figure 2.5.

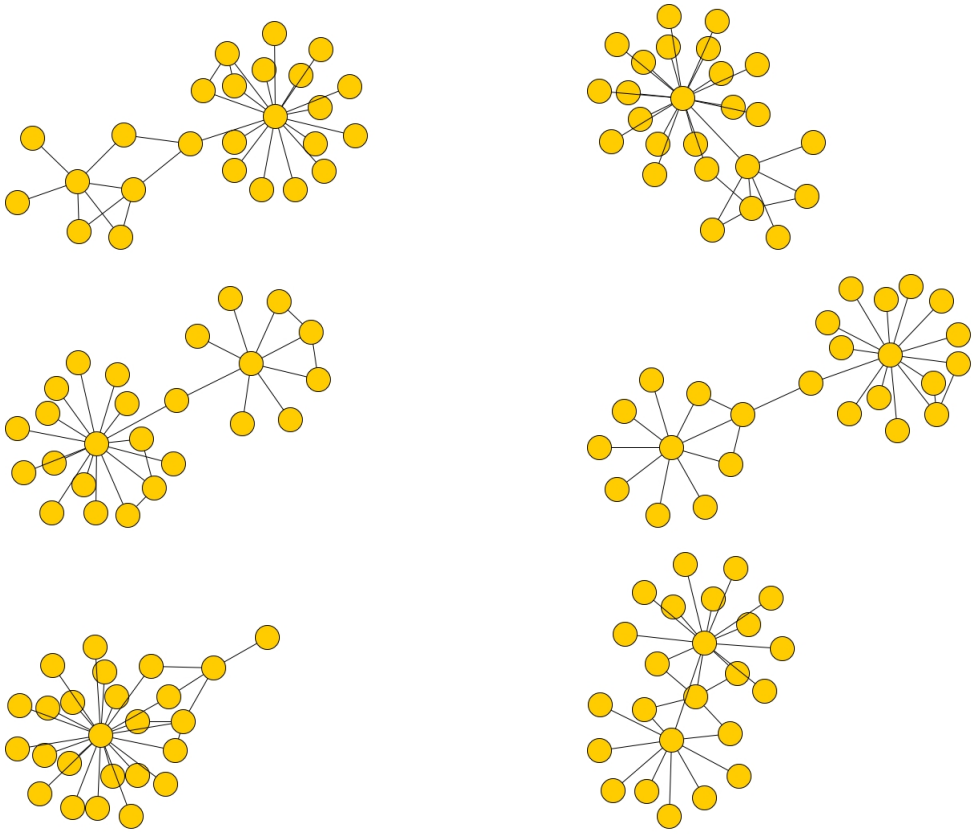


Figure 2.5: Some examples of subgraphs of G_0 used for learning our subgraph model.

Notice that the graphs in $\mathcal{G}$ do not have attributes, i.e. the vertex and edge spaces are:

$$\Lambda_V = V_{\text{ID}}$$

$$\Lambda_E = \{0, 1\},$$

where $V_{\text{ID}} = \{1, \ldots, 25\}$ just provides each vertex with a unique identification number. For modeling purposes, we introduce attributes for the vertices, which serve as latent variables; without them, very high-order templates would be necessary. We assign attributes to the vertices based on some simple vertex properties that can be used to distinguish some types

of vertices from others. Let the new vertex space be $\Lambda_V^{\text{new}} = \{c_1, c_2, c_3\} \times V_{\text{ID}}$, where each c_i denotes a color, and assign each vertex a color attribute based on its edge count. Let $d_G(v)$ denote the number of edges incident on vertex v in the graph G. Assign a color attribute to each vertex as follows:

$$\text{color}(v) = \begin{cases} c_1, \text{ if } 0 \leq d(v) \leq 1 \\ c_2, \text{ if } 2 \leq d(v) \leq 4 \\ c_3, \text{ if } 5 \leq d(v) \end{cases} .$$

For an example of this assignment, see figure 2.6. We note a couple of things: (1) more complicated attributes can be used for the vertices, for example using a generalized notion of edge counts; (2) a clustering algorithm can be used here, for example clustering similar subgraphs.

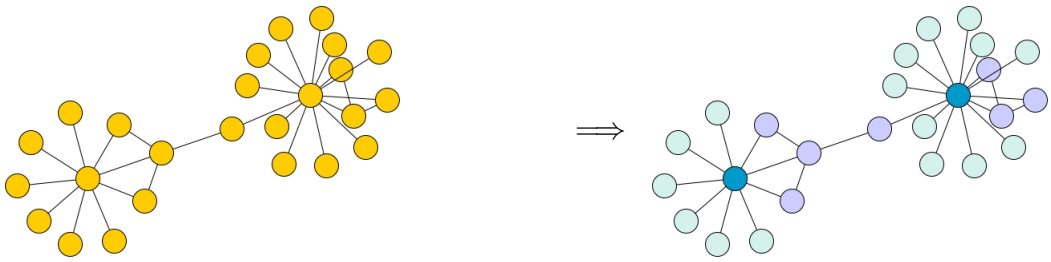


Figure 2.6: An example of a graph being assigned color attributes.

The templates used for our model are shown in tables 2.3, 2.4, and 2.5.

	λ_1
	λ_2
	λ_3

Table 2.3: The set of 1st-order graphs that are used as templates.

	λ_4
	λ_5
	λ_6
	λ_7
	λ_8
	λ_9

Table 2.4: The set of 2nd-order graphs that are used as templates.

	λ_{10}
	λ_{11}

Table 2.5: The set of 3rd-order graphs that are used as templates.

We learn model parameters using the algorithm in section 3.2, and a small training set of 31 graphs (examples shown in figure 2.5). Some samples from the model are shown in figure 2.7. As mentioned, these are only samples of subgraphs. With only one full graph

G_0, we cannot develop a hierarchical model, so instead, we combine these subgraphs just using a few simple rules (e.g., combining them by randomly placing edges between them). An example is shown in figure 2.8.

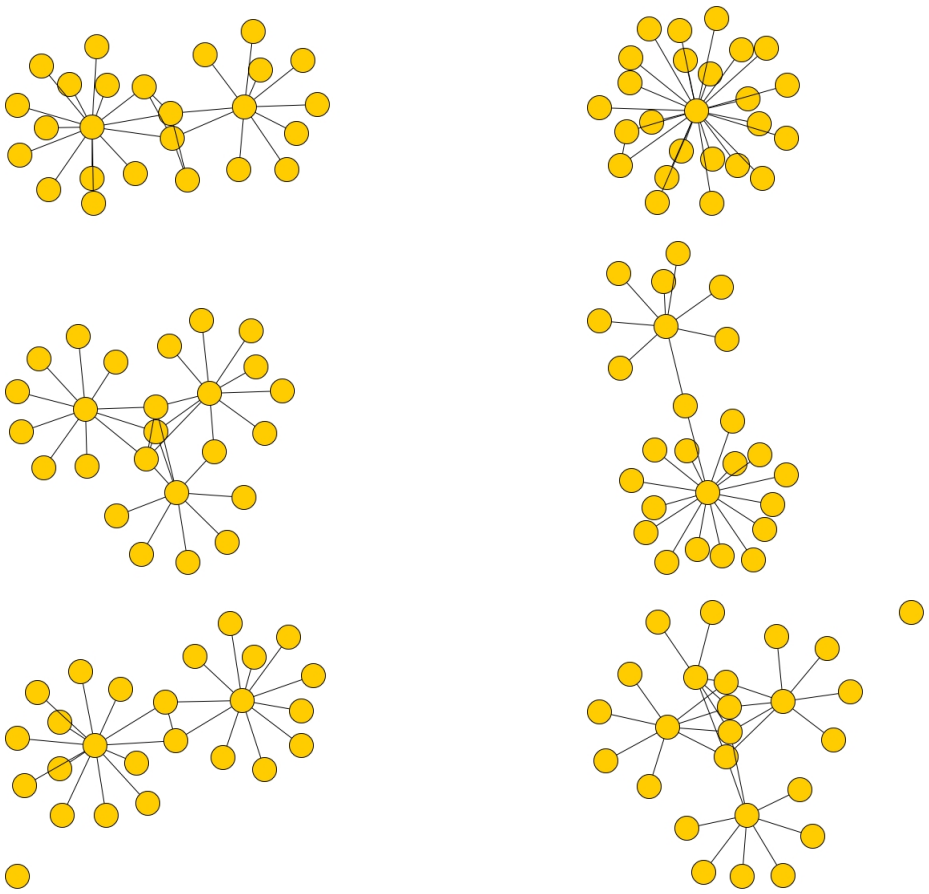


Figure 2.7: Model samples.

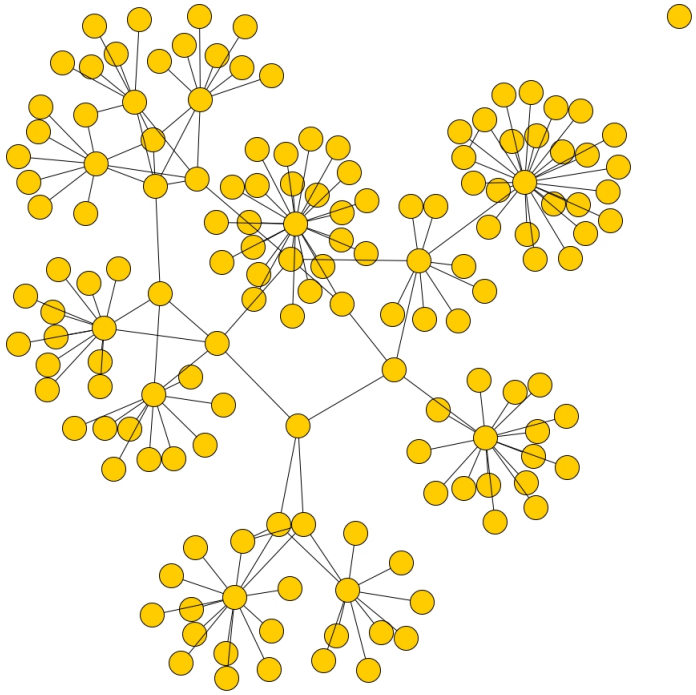


Figure 2.8: Mouse visual cortex sample from our model.

2.7.4 Example 4: Chemistry Data

In the chemoinformatics dataset MUTAG [Shervashidze et. al., 2011], there are 188 mutagenic aromatic and heteroaromatic nitro compounds. Examples are shown in figure 2.9.

We will form a simple hierarchical model using deterministic subgraphs. Define a set of subgraphs as shown in table 2.6. These subgraphs will correspond to vertices in the second level of the model. There will be an edge between two vertices in the second level if there is an intersection between two subgraphs. For example, let G be a molecule graph and suppose $G_1, G_2 \subset G$ are two subgraphs of G and are in table 2.6 below. If G_1, G_2 have a common subgraph (i.e. have at least one common vertex in G), then in the second level of the model, there will be an edge between the corresponding vertices. This edge will have an attribute specifying the degree of intersection, i.e. how many common vertices the two subgraphs share. See figure 2.10 for some examples. Thus, all the randomness in the problem is at this second level in the hierarchy and a model can be learned.

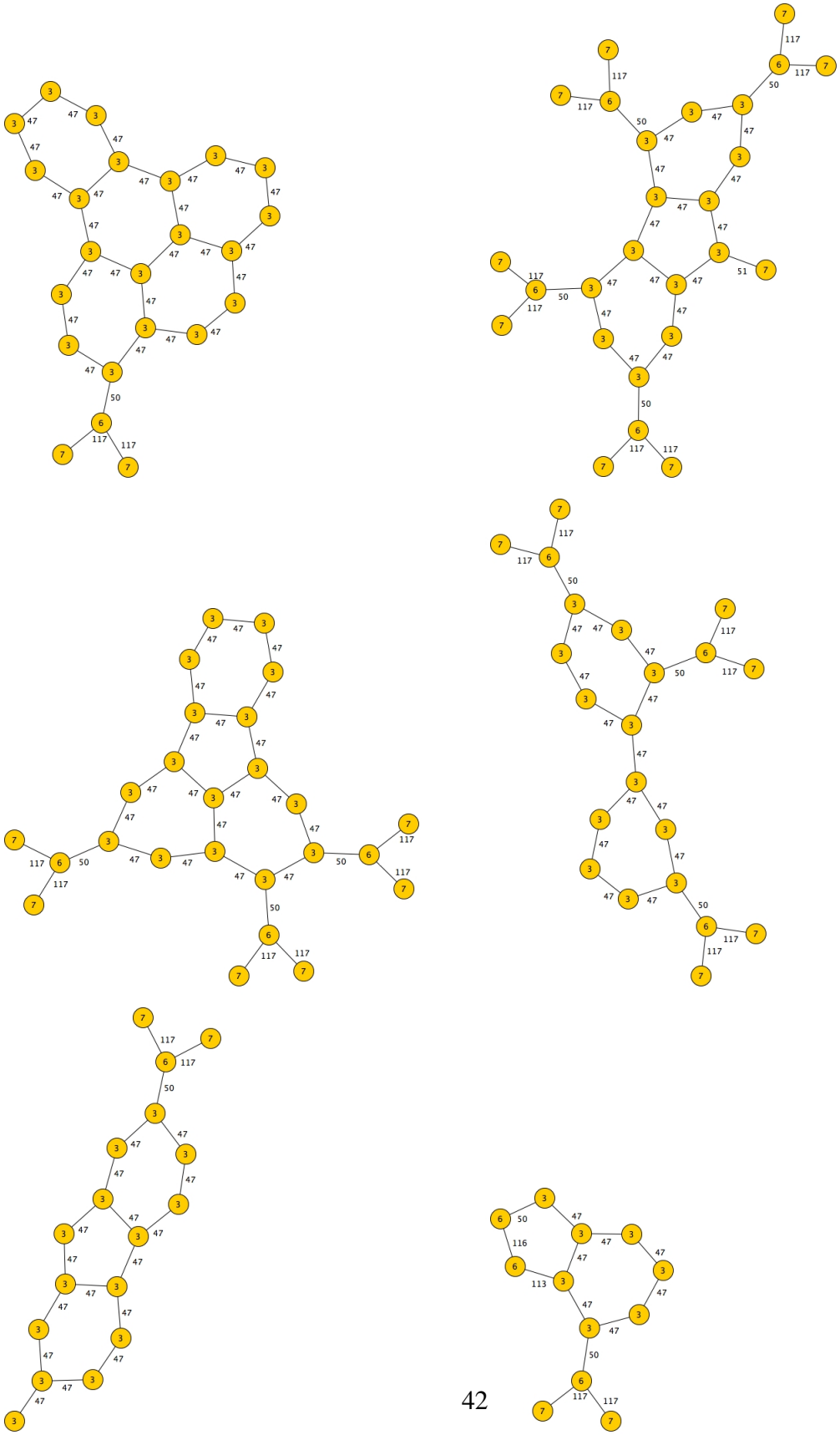


Figure 2.9: Examples of molecule graphs in the MUTAG dataset.

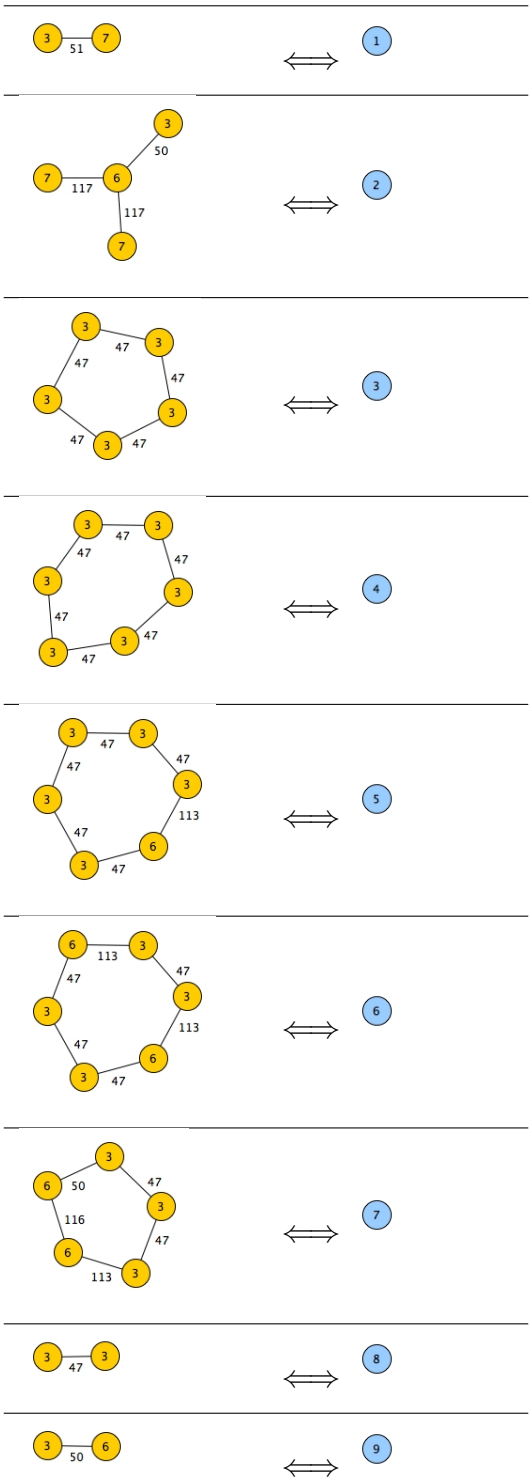


Table 2.6: Parts in the model. The blue vertices on the right-hand side represent the corresponding graph on the left-hand side. If one of the graphs on the left-hand side is a subgraph in a larger graph, then we may simplify the description of that larger graph through the use of these parts.

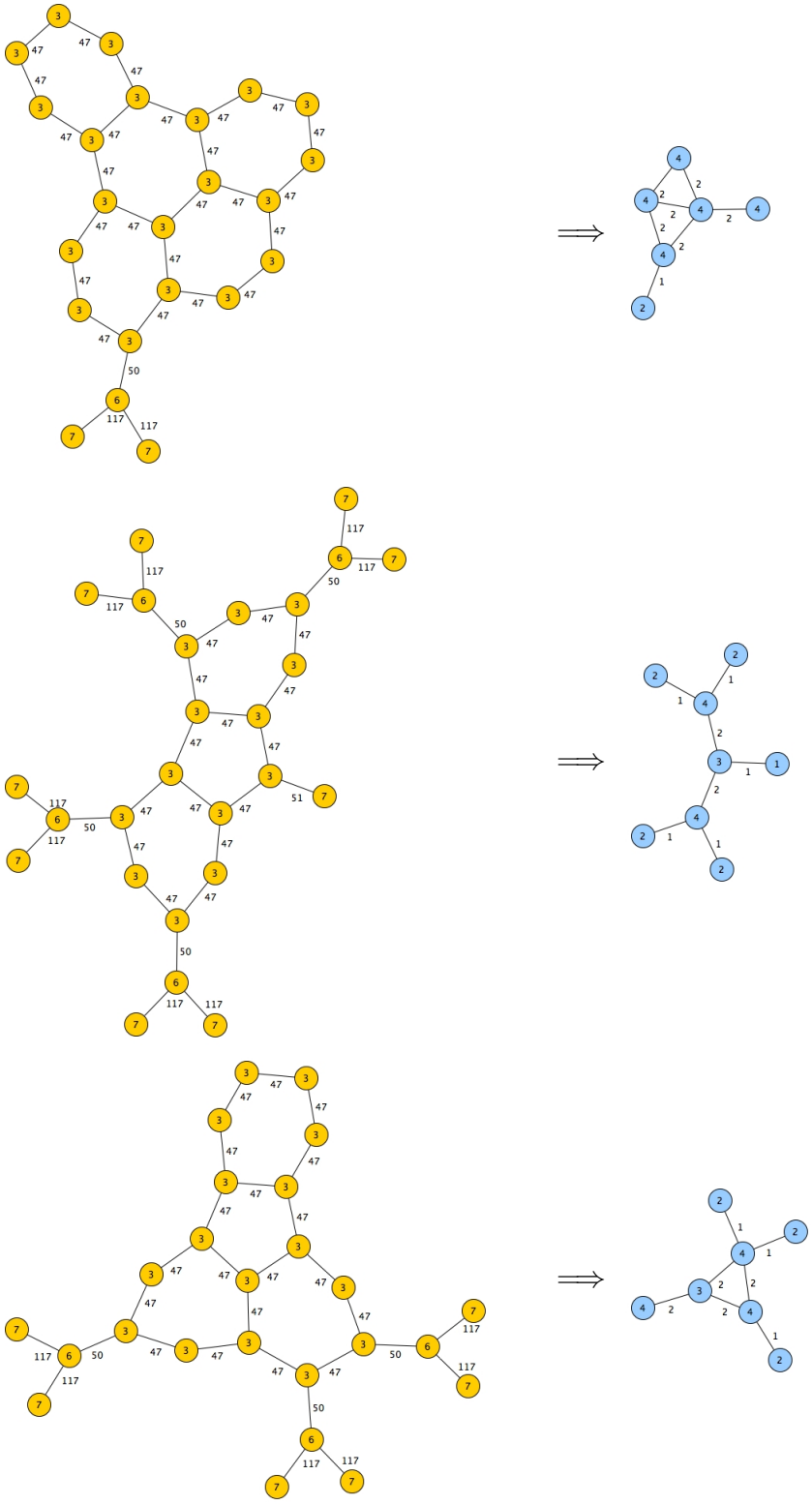


Figure 2.10: Examples of molecule graphs (from the MUTAG dataset) depicted by higher-level parts (subgraphs) rather than their lowest level parts. It should be easier to learn a distribution over this higher-level description of these graphs.

2.7.5 Example 5: Vertices with Color and Location

We now consider an example in which the vertex space has both a color and a location component. Let the vertex space be $\Lambda_V = \mathcal{C} \times \mathcal{L}$, where:

$$\mathcal{C} = \{c_1, c_2, c_3, c_4\}$$

$$\mathcal{L} = \{1, \ldots, p\} \times \{1, \ldots, p\}$$

and each c_i represents a color, and $\mathcal{L}$ represents a location space, a two dimensional grid of size p. Let the edge space be $\Lambda_E = \{0, 1\}$, representing the absence and presence of an edge, respectively.

We will define a master interactions function $U : \Lambda_V \times \Lambda_V \to \Lambda_E$ that assigns the value 0 to every pair of vertices that cannot have an edge, and assigns a value 1 to every pair of vertices that can have an edge. Define U as follows:

$$U(v, v') = \begin{cases} 0, & \text{if } d(v, v') > t \\ 1, & \text{otherwise} \end{cases}$$

where $d(v, v')$ is some distance function that assigns a distance between vertices based on their location attributes. In other words, this master interactions function can be used to ensure there is no edge between vertices that are farther apart than $t \in \mathbb{R}$. We will require

that no two vertices in a graph can have the same location, i.e.

$$\text{loc}(v) \neq \text{loc}(v') \text{ for all } v, v' \in V,$$

where $\text{loc}(v)$ is the projection of a vertex onto its location component. Let N be the maximum order of graphs in the graph space. Let the graph space be:

$$\mathcal{G} = \left\{ (V, E) \ : \begin{array}{l} V \subset \Lambda_V, \ |V| \leq N \\ E : V \times V \to \Lambda_E \\ E(v, v') \leq U(v, v') \text{ for all } v, v' \in V \\ \text{loc}(v) \neq \text{loc}(v') \text{ for all } v, v' \in V \end{array} \right\}.$$

The templates used are shown in figures 2.7, 2.8, and 2.9. For each template graph, we define a compatibility map $R : \mathcal{G} \to \{0, 1\}$ based on two relations $\{\sim_1, \sim_2\}$. These relations will ensure color consistency and spatial invariance respectively. Define them as follows:

1. **Color**: We want to consider a graph to be compatible with a template if the colors of the vertices match. Define a relation $\sim_1$ on $\mathcal{G}^{(1)}$ as follows:

$$G \sim_1 G' \iff \text{color}(v) = \text{color}(v')$$

where $\text{color}(v)$ is the projection of v to its first factor, $G, G' \in \mathcal{G}^{(1)}$, and v and v' is

each graphs vertex, respectively.

2. **Spatial & Structure Invariance**: We want to consider a graph to be compatible with a template if (a) the distances between the vertices are close to each other; and (b) the edge structure is the same. Define a relation $\sim_2$ on $\mathcal{G}^{(2)}$ as follows:

$$G \sim_2 G' \iff \begin{array}{l} |d(v_0, v_1) - d(v_0', v_1')| < \epsilon \\ E(v_0, v_1) = E'(v_0', v_1') \end{array}$$

where $d(v, v')$ denotes some distance function between the vertices based on their location attributes, $G, G' \in \mathcal{G}^{(2)}$, and $\{v_0, v_1\}$ and $\{v_0', v_1'\}$ is each graphs vertices, respectively.

Some samples are shown in figures 2.11. These were generated using the sampling algorithm in chapter 3.

template G_k	template locations	parameter value λ_k
	$(x, y) \in \{1, \ldots, p\}^2$	$\lambda_{1,x,y} = 0.5$
	$(x, y) \in \{1, \ldots, p\}^2$	$\lambda_{2,x,y} = 0.4$
	$(x, y) \in \{1, \ldots, p\}^2$	$\lambda_{3,x,y} = 0.5$
	$(x, y) \in \{1, \ldots, p\}^2$	$\lambda_{4,x,y} = 0.4$

Table 2.7: The set of 1st-order graphs that are used as templates. There are $(4 \times p^2)$ templates of 1st-order being used here. Parameters were hand-tuned here.

template G_k	parameter value λ_k	template G_k	parameter value λ_k
	$\lambda_{5,a} = 0.5$		$\lambda_{5,b} = 0.5$
	$\lambda_{6,a} = 1.5$		$\lambda_{6,b} = 1.5$
	$\lambda_{7,a} = 0.4$		$\lambda_{7,b} = 0.4$
	$\lambda_{8,a} = 1.5$		$\lambda_{8,b} = 1.5$
	$\lambda_{9,a} = -\infty$		$\lambda_{9,b} = -\infty$
	$\lambda_{10,a} = -\infty$		$\lambda_{10,b} = -\infty$
	$\lambda_{11,a} = -\infty$		$\lambda_{11,b} = -\infty$
	$\lambda_{12,a} = -\infty$		$\lambda_{12,b} = -\infty$
	$\lambda_{13,a} = -\infty$		$\lambda_{13,b} = -\infty$
	$\lambda_{14,a} = -\infty$		$\lambda_{14,b} = -\infty$

Table 2.8: The set of 2nd-order graphs that are used as templates. Parameters were hand-tuned here.

template G_k	parameter value λ_k	template G_k	parameter value λ_k
	$\lambda_{15,a} = -5$		$\lambda_{15,b} = -5$
	$\lambda_{16,a} = -5$		$\lambda_{16,b} = -5$
	$\lambda_{17} = -3$		
	$\lambda_{18,a} = 0.75$		$\lambda_{18,b} = 0.75$
	$\lambda_{19,a} = -5$		$\lambda_{19,b} = -5$
	$\lambda_{20,a} = -\infty$		$\lambda_{20,b} = -\infty$
	$\lambda_{21,a} = -\infty$		$\lambda_{21,b} = -\infty$
	$\lambda_{22,a} = -\infty$		$\lambda_{22,b} = -\infty$
	$\lambda_{23,a} = -\infty$		$\lambda_{23,b} = -\infty$

Table 2.9: The set of 3rd-order graphs that are used as templates. Parameters were hand-tuned here.

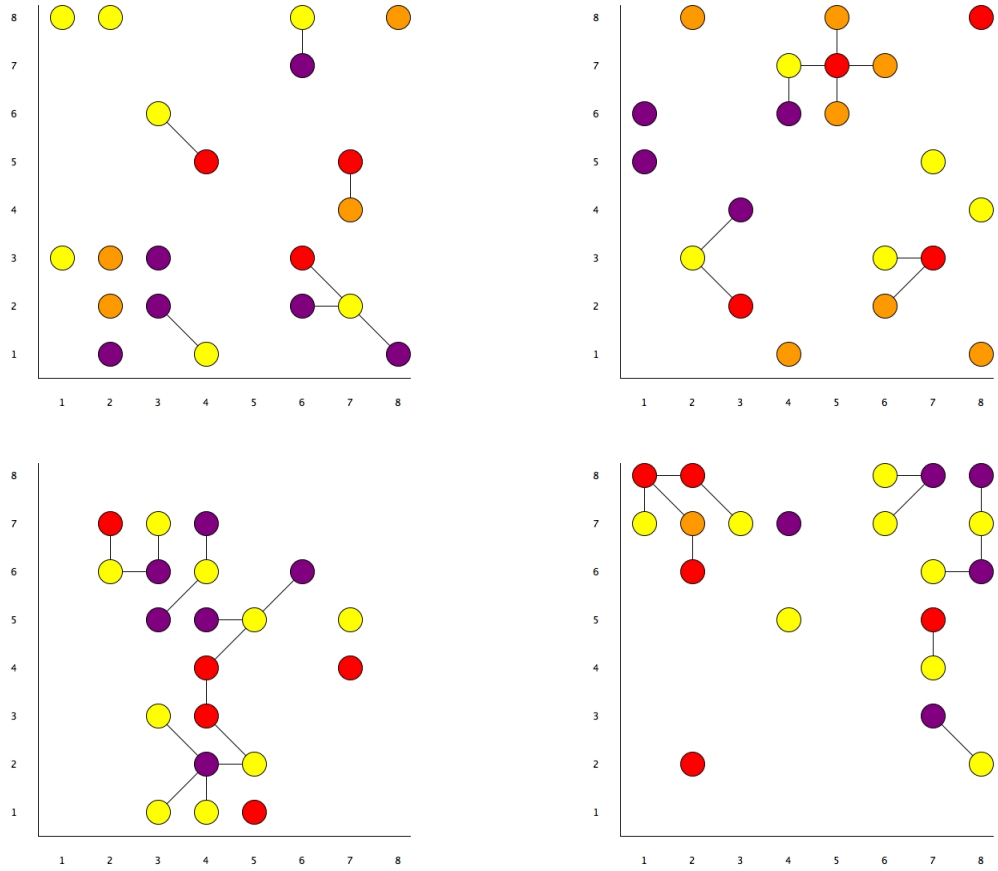


Figure 2.11: Samples from the model.

2.8 Discussion

In this chapter, in order to define a Gibbs distribution for random graphs, we used a projection on graph spaces (i.e., projection A). In this section, we define another projection (i.e., projection B) that may also be used for this purpose and has some other properties that may be valuable, e.g., can be used to also define marginal random graphs. It is not yet clear to us if one projection should be preferred over the other; this might be resolved,

however, with some further research into random graphs (e.g., by investigating definitions of a Markov distribution). Define this alternative projection as follows:

Definition: (Projection B) Let $V \subseteq \Lambda_V$ be a set of vertices. Define the projection $\tilde{\pi}_V : \mathcal{G} \to \tilde{\mathcal{G}}_V$, where $\tilde{\mathcal{G}}_V = \{G \in \mathcal{G} \mid V(G) \subseteq V\}$, as follows:

$$\tilde{\pi}_V(G) = G(V \cap V'),$$

where $V' = V(G)$ is the set of vertices of the graph $G \in \mathcal{G}$.

The projection $\tilde{\pi}_V$ maps graphs to their induced subgraphs based on the intersection of their vertices with the vertices V. That is, for a graph G, if there are no vertices in this intersection (i.e., $V \cap V' = \emptyset$, where $V' = V(G)$), then G gets projected to the empty graph; if there is an intersection (i.e. $V \cap V' \neq \emptyset$, where $V' = V(G)$), then G gets projected to its subgraph induced by the vertices in this intersection.

This projection has the property that the image of a projectable graph space is also a projectable space. That is, if the domain $\mathcal{G}$ is projectable, then for each projection $\tilde{\pi}_V$, the codomain $\tilde{\mathcal{G}}_V \subseteq \mathcal{G}$ is also projectable. This property is useful because it allows us to define a consistent set of marginal random variables. Suppose we have a distribution P over the graph space $\mathcal{G}$; the distribution for a marginal random variable $\mathbf{G}_V$ taking values in $\tilde{\mathcal{G}}_V$ is

defined as:

$$P_V^{\text{marg}}(G) = \sum_{\substack{G' \in \mathcal{G} \\ \tilde{\pi}_V(G') = G}} P(G'), \quad G \in \tilde{\mathcal{G}}_V.$$

It can be verified that this defines a valid probability distribution, i.e., that

$$\sum_{G \in \tilde{\mathcal{G}}_V} P_V^{\text{marg}}(G) = 1,$$

and further, that this set of marginal random variables (i.e., the set $\{\mathbf{G}_V, V \subseteq \Lambda_V\}$) is consistent. (In contrast, for projection A, the codomains are not projectable graph spaces, and this projection does not produce a consistent set of marginal random variables.)

These marginal random variables for random graphs allow us to use the standard definitions of independence and conditional independence for random variables. In future work, we plan on investigating dependency structures, e.g., Markov properties for random graphs, and their connections to Gibbs distribution.

Chapter 3

Inference & Learning

In this chapter, we discuss inference for random graphs; for a given probability distribution, inference refers to the calculation (or estimation) of probabilities in that distribution, or more generally, of functions of probabilities in that distribution. Inference can be performed by sampling from distributions; a standard Metropolis-Hastings algorithm is presented here for this purpose.

Next, we present a learning algorithm for random graph models; for a given model, learning refers to the selection of a particular distribution in it. A stochastic learning algorithm is presented here.

3.1 Sampling

Suppose we have a vertex space Λ_V and an edge space Λ_E, and let $\mathcal{G}$ be a finite graph space with respect to them. Further, suppose we have a distribution P over $\mathcal{G}$ that we want to sample from. We will use Markov Chain Monte Carlo (MCMC), and in particular, the Metropolis-Hastings algorithm:

Algorithm: (Metropolis-Hastings) Given a *transition kernel* $q(G'|G)$ and starting from an initial state G_1, repeat the following steps from $t = 1$ to T:

1. Generate a candidate $G' \sim q(G'|G_t)$.

2. Generate $U \sim \mathcal{U}(0, 1)$ and set

$$G_{t+1} = \begin{cases} G', & \text{if } U \leq \alpha(G_t, G') \\ G_t, & \text{otherwise} \end{cases}$$

where $\alpha(G, G')$ is the *acceptance probability*, given by:

$$\alpha(G, G') = \min \left\{ \frac{P(G')\ q(G\ |G')}{P(G)\ q(G'\ |G)}, 1 \right\}.$$

This algorithm will generate a sequence $G_1, G_2, \ldots$ of dependent random graphs, and for large t, the graph G_t will be approximately distributed according to P (assuming an appropriate transition kernel). Given a graph $G = (V, E)$, the transition kernel will generate

a proposal graph G' based on simple moves. There are many possibilities, but for most problems, we find the following moves to be sufficient:

1. Adding a vertex $v \in \Lambda_V$ to V (and not adding any edges, i.e., $E(v, \cdot) = 0$.)

2. Deleting a vertex $v \in V$ from V (from among vertices with no incident edges).

3. Changing the value of an edge in E (i.e., changing the value of $E(v, v') \in \Lambda_E$ for some $v, v' \in V$).

The probability of each type of move can be uniform, although sometimes non-uniform probabilities may be preferable (e.g., assigning a greater probability to edge moves). Each move in this set has an inverse move allowing a chain to return to the previous state; hence these moves satisfy the weak symmetry property. We consider sampling from conditional distributions in chapter 6.

3.1.1 Computation

We now consider computational efficiencies for this sampling algorithm. Suppose we have a distribution P in the form of equation (2.3), i.e.:

$$P(G) = \frac{1}{Z} \exp\left[\sum_{k=1}^{K} \lambda_k U_k(G)\right],$$

and define H as the exponent in this distribution:

$$H(G) = \sum_{k=1}^{K} \lambda_k U_k(G).$$

In this section, we consider the calculation of differences of the form

$$\triangle H \equiv H(G_{\text{new}}) - H(G_{\text{old}}),$$

where G_{new} and G_{old} are graphs. In the naive approach, each exponent is computed separately, and the difference taken. However, the difference $\triangle H$ can be calculated efficiently by ignoring subgraphs that are shared between the two graphs. Define $J_0 \subseteq S(G_{\text{new}})$ as the set of subgraphs of G_{new} that are not subgraphs of G_{old}; similarly, define $J_1 \subseteq S(G_{\text{old}})$ as the set of subgraphs of G_{old} that are not subgraphs of G_{new}. That is:

$$J_0 \equiv \{G' \in S(G_{\text{new}}) \mid G' \notin S(G_{\text{old}})\} \cap \mathcal{G}_{\text{basis}}$$

$$J_1 \equiv \{G' \in S(G_{\text{old}}) \mid G' \notin S(G_{\text{new}})\} \cap \mathcal{G}_{\text{basis}}.$$

Hence, we have that

$$\begin{aligned}\triangle H &\equiv H(G_{\text{new}}) - H(G_{\text{old}}) \\ &= \sum_{k=1}^{K} \lambda_k \left[U_k(G_{\text{new}}) - U_k(G_{\text{old}})\right] \\ &= \sum_{k=1}^{K} \lambda_k \left[\tilde{U}_k(G_{\text{new}}) - \tilde{U}_k(G_{\text{old}})\right]\end{aligned}$$

where

$$\tilde{U}_k(G_{\text{new}}) = \#\{G' \in J_0 \ : \ R_k(G') = 1\}$$

$$\tilde{U}_k(G_{\text{old}}) = \#\{G' \in J_1 \ : \ R_k(G') = 1\}.$$

Let's consider some examples.

New Node: Let G_{old} be any graph and suppose we formed G_{new} by adding a vertex u to it (and possibly edges). In this case, we have that $G_{\text{old}} \subset G_{\text{new}}$ (i.e. is an induced subgraph) and hence:

$$J_0 = \{G' \in S(G_{\text{new}}) \mid u \in V(G')\} \cap \mathcal{G}_{\text{basis}}$$

$$J_1 = \emptyset.$$

Deleted Node: Let G_{old} be a graph (with at least one vertex) and suppose we formed G_{new}

by deleting a vertex $u \in V(G_{\text{old}})$. In this case, we have that $G_{\text{new}} \subset G_{\text{old}}$ and hence:

$$J_0 = \emptyset$$

$$J_1 = \{G' \in S(G_{\text{old}}) \mid u \in V(G')\} \cap \mathcal{G}_{\text{basis}}.$$

New Edge: Let G_{old} be a graph (with at least two vertices) and suppose we formed G_{new} by changing the value of an edge $E(v, v')$ for some $v, v' \in V(G_{\text{old}})$. In this case, we have that:

$$J_0 = \{G' \in S(G_{\text{new}}) \mid v, v' \in V(G')\} \cap \mathcal{G}_{\text{basis}}$$

$$J_1 = \{G' \in S(G_{\text{old}}) \mid v, v' \in V(G')\} \cap \mathcal{G}_{\text{basis}}.$$

3.2 Learning

To estimate the parameters $\lambda = \{\lambda_1, \ldots, \lambda_K\}$ in the model in equation (2.3), we use the maximum likelihood estimate (MLE). Suppose we have a set of graphs $\{G_1, G_2, \ldots, G_N\}$ sampled according to $P(G; \lambda^*)$, where $\lambda^* \in \Lambda$ and where $\Lambda \subset \mathbb{R}^K$ is a compact set. The MLE is the solution to the following optimization problem:

$$\hat{\lambda} = \arg\max_{\lambda \in \Lambda} \prod_{i=1}^{N} P(G_i; \lambda).$$

It can be shown that the MLE is also a solution to the equations:

$$\mathbb{E}_\lambda[U_k(G)] = \hat{U}_k, \quad k = 1, \ldots, K,$$

where $\hat{U}$ are the empirical statistics of the features:

$$\hat{U}_k = \frac{1}{N} \sum_{i=1}^{N} U_k(G_i), \quad k = 1, \ldots, K.$$

We run a stochastic approximation algorithm ([Younes, 1988], [Salakhutdinov, 2009]) for estimating the solution (see Algorithm 1 below). This stochastic algorithm uses multiple Markov chains in parallel; we find this beneficial in practice since individual chains can sometimes become stuck in certain regions for long periods of time.

Algorithm 1 Stochastic Approximation Procedure

Input:

Empirical statistics $\hat{U}$;

Initial parameters λ^1;

Initial set of M particles $\{G^{1,1}, \ldots, G^{1,M}\}$;

for $t = 1 : T$ (number of iterations) **do**

for $m = 1 : M$ (number of parallel Markov chains) **do**

Sample $G^{t+1,m}$ given $G^{t,m}$ using the transition operator $T_{\lambda^t}(G^{t+1,m}, G^{t,m})$

end for

end for

Update: $\lambda^{t+1} = \lambda^t + \alpha_t \left[\hat{U} - \frac{1}{M}\sum_{m=1}^{M} U(G^{t+1,m})\right]$

Decrease α_t

Chapter 4

Graph Discovery

There are often situations in which there is an unknown graph that one wants to determine. *Graph discovery* refers to the problem of uncovering these unknown graphs by sequential questioning: the answer to any question, when asked about a graph, provides information about it. For questions to be efficient, we must have some knowledge of the likelihoods of graphs; assume we have a distribution over the graph space. (This distribution can be acquired, for example, using the statistical models from chapter 2). Then, questions can be selected that maximize the expected information about the unknown graph, given the previous questions and answers. In this chapter, we consider some useful question spaces, discuss the important concept of vertex instantiation, and consider question sequences and criteria for their generation.

4.1 Questions

Let a question be a function of the form $f : \mathcal{G} \to \mathbb{A}$, taking graphs from some graph space $\mathcal{G}$ to answers in some answer space $\mathbb{A}$. This answer space, in general, can be complicated and depends on the type of questions we want to employ. However, suppose there is a cost associated with each question, and moreover, let this cost be proportional to the cardinality of the answer space $|\mathbb{A}|$. With this assumption, the most efficient[1] questions are always binary (i.e. questions with an answer space $\mathbb{A} = \{0, 1\}$) or ternary; we consider binary questions in this work.

4.2 Simple Questions

In this section, we describe questions that can be used for identifying vertices in a graph. In the next section, we consider a more sophisticated version of these questions. Let Λ_V be the vertex space (associated with the graph space $\mathcal{G}$) and let $B \subset \Lambda_V$ be a subset. Now, define a *counting function* $N_B : \mathcal{G} \to \{0, 1, 2, \ldots\}$ as a function that takes a graph G and returns the number of its vertices that exist in B. That is, let

$$N_B(G) = \sum_{v \in V} I_{\{v \in B\}}, \text{ where } B \subset \Lambda_V.$$

[1] This follows since questions are selected sequentially. Notice, that any answer space can be decomposed into a binary representation, which in turn corresponds to a set of binary questions. Loosely speaking, for any non-binary question, it is always preferential to ask one of its corresponding binary questions and decide the next question to ask in the context of this additional information. We note that if we base the cost of questions on other criteria, such as the effort necessary by a human to answer it, the use of only binary questions may then be highly inefficient.

Now, we define questions that can establish the existence of vertices in the unknown graph.

1. Existence questions: A question f_B is an *existence question* if it has the form:

$$f_B(G) = I_{\{N_B(G)>0\}} \text{ where } B \subset \Lambda_V.$$

An existence question is positive if the graph G has one or more of its vertices in B. Let the set of all existence questions be denoted by $\tilde{\mathcal{F}}_{\text{exist}}$.

2. Uniqueness questions: A question f_B is an *uniqueness question* if it has the form:

$$f_B(G) = I_{\{N_B(G)=1\}} \text{ where } B \subset \Lambda_V.$$

A uniqueness question is positive if the graph G has one and only one of its vertices in B. Notice that for any uniqueness question f_B, if a graph G receives a positive answer (i.e. $f_B(G) = 1$), a vertex in this graph can be uniquely identified; the set B distinguishes this vertex from all others in the graph. We consider the important idea of vertex instantiation in the next section. Let the set of all uniqueness questions be denoted by $\tilde{\mathcal{F}}_{\text{uniq}}$.

Example 4.1: This will be a running example in this chapter. Suppose we have a vertex

space $\Lambda_V = \mathcal{C} \times \mathcal{L}$, where:

$$\mathcal{C} = \{c_1, c_2\}$$

$$\mathcal{L} = \{1, \ldots, p\} \times \{1, \ldots, p\}$$

and each c_i represents a color, and $\mathcal{L}$ represents a location space, a two dimensional grid of size p. Let the edge space be $\Lambda_E = \{0, 1, 2\}$, where the value 0 represents the absence of an edge. Suppose the graph G shown in figure 4.1 is an unknown graph that we want to determine. Let $B \subset \Lambda_V$ be the subset containing all vertices that have a location component in the orange rectangle depicted in the figure (i.e., $B = \{v \in \Lambda_V \mid \text{loc}(v) \in R\}$, where $R \subset \mathcal{L} = \{1, \ldots, p\}^2$ is the subset of locations depicted by the rectangle, and $\text{loc}(v)$ denotes the projection of v to its location component). Then, for an existence question f_B, we have that $f_B(G) = I_{\{N_B(G)>0\}} = 1$ since the graph has a vertex in B, and for a uniqueness question f_B, we have that $f_B(G) = I_{\{N_B(G)=0\}} = 1$ since the graph has one and only one vertex in B. Thus, this latter question uniquely identifies this vertex from the others in the graph.

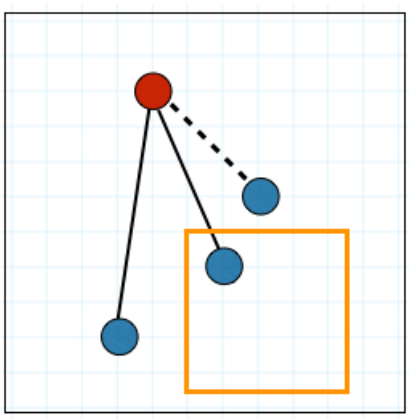


Figure 4.1: An example graph and a designated region (the orange rectangle) used in the questions in example 4.1.

We define the set of instantiation questions as $\tilde{\mathcal{F}}_{\text{inst}} \equiv \tilde{\mathcal{F}}_{\text{exist}} \cup \tilde{\mathcal{F}}_{\text{uniq}}$. In practice, there are limits on the set of questions that can be used; for example, instantiation questions may be uninterpretable for many $B \subset \Lambda_V$ (uninterpretable in the sense that there does not exist a compact description of the question). Thus, assume we only use instantiation questions that involve $B \in \mathcal{B}$, where $\mathcal{B} \subset \mathbb{P}(\Lambda_V)$ is some collection of allowable sets. We denote this set of questions by:

$$\tilde{\mathcal{F}}_{\text{inst}}(\mathcal{B}) \equiv \{f_B \in \tilde{\mathcal{F}}_{\text{inst}} \mid B \in \mathcal{B}\}.$$

4.3 Instantiation

Instantiation of a vertex refers to: (1) the discovery of some set $B \subset \Lambda_V$ that distinguishes that vertex from all others in the unknown graph; and (2) the denotation of this unique vertex with a label (e.g. vertex v_i). Let $G = (V, E)$ be an unknown graph and

suppose we have asked a sequence of instantiation questions $F_k = (f_1, \ldots, f_k)$ about it. Denote the set of vertices instantiated by these questions by

$$\Omega^{\text{Inst}}(G, F_k) \subset V,$$

and denote the set of vertices that have not been instantiated by

$$\Omega^{\neg\text{Inst}}(G, F_k) = V \setminus \Omega^{\text{Inst}}(G, F_k).$$

Furthermore, assume the set $\Omega^{\text{Inst}}(G, F_k)$ is ordered: let it retain the order in which vertices have been instantiated. For example, these ordered sets have the form:

$$\Omega^{\text{Inst}}(G, F_k) = (v_1, \ldots, v_{n'}),$$

where each $v_i \in V$ and the index indicates this vertex was the i^{th} vertex to be instantiated. Although a vertex $v \in \Omega^{\text{Inst}}(G, F_k)$ has been instantiated, in general, it is not completely known: we only know that it exists in some subset $B \subset \Lambda_V$. Finally, we mention that, more generally, subgraphs can be instantiated[2].

Since every vertex in a graph is unique (by our definition of a graph), every vertex can theoretically be instantiated. However, since in practice we only use a subset of the possible

[2]Instantiation of a subgraph of order k refers to: (1) the discovery of some set $B \subset \mathcal{G}^{(k)}$ that distinguishes that subgraph from all others in the unknown graph; and (2) the denotation of this unique subgraph with a label (e.g. subgraph $G' \subset G$). Notice that once some set of vertices is instantiated, so too is the induced subgraph that corresponds to them.

questions, we must ensure question spaces have enough capacity to do so. For example, suppose we try to instantiate vertices using questions about location (assuming the vertices have a location component), and we only use a coarse set of rectangular windows; then, if a graph contains many vertices that are located near each other, it may not be possible to instantiate them with these questions. We now consider this topic.

4.4 Complex Questions

Thus far, we have described a set of questions $\tilde{\mathcal{F}}_{\text{inst}}$ that can be relatively small and, loosely speaking, has a somewhat limited capacity to differentiate graphs. To increase this capacity, we now consider a larger space of questions, one with questions that can be directly applied to unexplored regions of an unknown graph. For example, we may want questions that can: (1) refine our knowledge of vertices that have already been instantiated, as well as the edges between them; or (2) focus on discovering new vertices that have not been instantiated. We now describe these types of questions more formally.

4.4.1 Post-Instantiation Questions

Suppose we have a graph G and a sequence of questions $F_k = (f_1, \ldots, f_k)$, and have instantiated a set of vertices:

$$V' = \Omega^{\text{Inst}}(G, F_k)$$
$$= (v_1, \ldots, v_{n'}).$$

As mentioned above, we only have a partial knowledge of these instantiated vertices. Suppose we want to learn more about them. This may be done by focusing attention on the subgraph $G' = G(V')$, i.e. the graph induced by the instantiated vertices. Let's consider some examples of post-instantiation questions.

1. Vertex refinement questions: Often, we want to focus attention on a single vertex, say $v_i \in V'$, the i^{th} instantiated vertex. This may be done by forming the trivial subgraph $G' = G(\{v_i\})$; here the counting functions take the simple form:

$$N_B(G') = I_{\{v_i \in B\}}, \text{ where } B \subset \Lambda_V.$$

We may now define questions for this vertex: a question $f_{B,F_k,i}$ is a *vertex refinement*

question if it has the following form:

$$f_{B,F_k,i}(G) = I_{\{N_B(G')=1\}}$$
$$= I_{\{v_i \in B\}},$$

where $G' = G(\{v_i\})$. Since we already have a partial knowledge of the vertex v_i, for example, that it exists in $B' \subset \Lambda_V$, we will want to ask vertex refinement questions in which $B \subset B'$. If the question is using a value i that exceeds the number of vertices that have been instantiated (i.e. $i > n'$), then the induced graph $G'' = G(\emptyset)$ is empty and $f_{B,F_k,i}(G) = 0$. Let the set of all vertex refinement questions be denoted by $\mathcal{F}_{\text{ref}(1)}$.

2. Subgraph refinement questions: Suppose we want to learn about the edge value $E(v_i, v_j)$ between $v_i, v_j \in V'$, the i^{th} and j^{th} instantiated vertices. This may be done by forming the subgraph $G' = G(\{v_i, v_j\})$ and asking if G' exists in some set $B \subset \mathcal{G}^{(2)}$. For example, suppose we want to know if the edge between v_i and v_j has some value $e \in \Lambda_E$. To do this, let

$$B = \{G \in \mathcal{G}^{(2)} \mid E(v, v') = e\},$$

where v and v' are the two vertices of graph G. More generally, let $I \subset \{1, \ldots, n'\}$ and

$B \subset \mathcal{G}^{(|I|)}$. A question $f_{B,F_k,I}$ is a *subgraph refinement question* if it has the form:

$$f_{B,F_k,I}(G) = I_{\{N_B(G')=1\}}$$
$$= I_{\{G' \in B\}},$$

where $G' = G(\{v_i, i \in I\})$. Let the set of all refinement questions for subgraphs of order k be denoted by $\mathcal{F}_{\text{ref}(k)}$.

Example 4.2: Let the setup be as in the previous example. Suppose we ask a sequence of questions F_k, and this sequence contains the uniqueness question f_B, where $B \subset \Lambda_V$ is the set of vertices contained in the orange rectangle in figure (4.1). Then, this sequence instantiates the vertex v_1 inside this rectangle (i.e., we have $v_1 \in \Omega^{\text{Inst}}(G, F_k)$). Now, we might want to refine our knowledge about this vertex, for example learn about of its color. Let $B' = \{v \in \Lambda_V \mid \text{color}(v) = \text{'red'}\}$, where color$(v)$ denotes the projection of the vertex v to its color component. The vertex refinement question $f_{B',F_k,1}(G) = 0$ in this case, since the vertex v_1 is not 'red', where we are assuming that v_1 was the first vertex instantiated.

4.4.2 Pre-Instantation Questions

Now suppose instead of focusing attention on instantiated vertices, we want to gather information about non-instantiated vertices. For example, it is often important to instantiate new vertices and this can be done more efficiently by removing from consideration vertices

that have already been instantiated. Suppose we have asked a sequence of questions $F_k = (f_1, \ldots, f_k)$ and we want to gain information about the non-instantiated vertices:

$$V' = \Omega^{\neg \text{Inst}}(G, F_k) \subset V.$$

This may be done by forming a new graph $G' = G(V')$, i.e. the graph induced by the non-instantiated vertices. Recall, above we defined existence and uniqueness questions as functions of the form:

$$f_B(G) = I_{\{N_B(G)>0\}}$$
$$f_B(G) = I_{\{N_B(G)=1\}},$$

where $B \subset \Lambda_V$. Now, we define a more sophisticated version; let existence and uniqueness questions be functions of the form:

$$f_{B,F_k}(G) = I_{\{N_B(G')>0\}}$$
$$f_{B,F_k}(G) = I_{\{N_B(G')=1\}},$$

where $G' = G(V')$ and $V' = \Omega^{\neg \text{Inst}}(G, F_k)$. We will denote this new set of existence and uniqueness questions by $\mathcal{F}_{\text{exist}}$ and $\mathcal{F}_{\text{uniq}}$, respectively, and let $\mathcal{F}_{\text{inst}} \equiv \mathcal{F}_{\text{exist}} \cup \mathcal{F}_{\text{uniq}}$. As above, we let $\mathcal{F}_{\text{inst}}(\mathcal{B})$ denote the set of instantiation questions using $B \in \mathcal{B}$, where $\mathcal{B} \subset \mathbb{P}(\Lambda_V)$ is some collection of allowable sets.

Example 4.3: Let the setup be as in the previous example, and as before, suppose we ask a sequence of questions F_k containing the uniqueness question f_B, where $B \subset \Lambda_V$ is the set of vertices contained in the orange rectangle on the left-hand side of figure 4.2. Then, this sequence instantiates the vertex v_1 in this rectangle (i.e., we have $v_1 \in \Omega^{\text{Inst}}(G, F_k)$). Suppose this is the only vertex instantiated by the sequence, and to instantiate more vertices, we ask another uniqueness question. Let $C \subset \Lambda_V$ be the set of vertices contained in the orange rectangle on the right-hand side of figure 4.2. Then, for the uniqueness question $f_{C,F_k} \in \mathcal{F}_{\text{uniq}}$, we have that $f_{C,F_k}(G) = I_{\{N_C(G')=1\}} = 1$, and we can instantiate the vertex to the left of vertex v_1. Notice, however, if we instead ask the uniqueness question $f_C \in \tilde{\mathcal{F}}_{\text{uniq}}$, then we have that $f_C(G) = I_{\{N_C(G)=1\}} = 0$, and this other vertex is not instantiated. This example illustrates one difference between these two question spaces; we now elaborate on the differences.

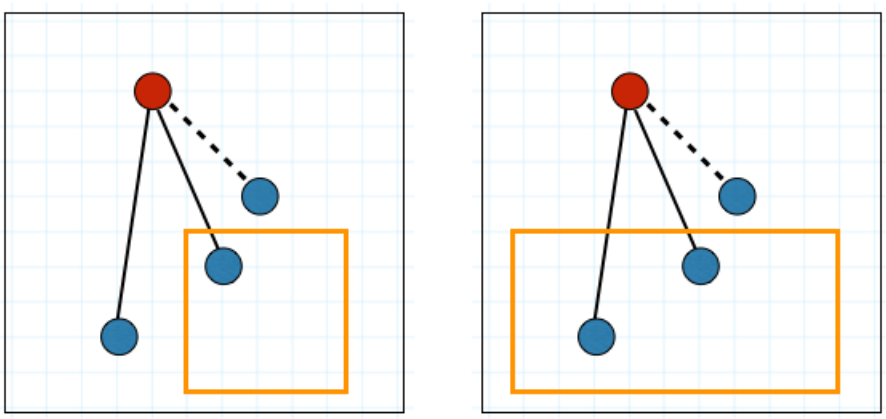


Figure 4.2: An example graph. The designated region in the left-hand figure can be used to instantiate the vertex in it. The designated region in the right-hand figure can also be used to instantiate a vertex, given that one of the two vertices in the region is already instantiated. See example 4.3.

So, how do these two versions of instantiation questions (i.e. $\mathcal{F}_{\text{inst}}$ and $\tilde{\mathcal{F}}_{\text{inst}}$) compare to each other? We discuss two important differences. The first difference between the two concerns the uniqueness of instantiations; when using questions in $\tilde{\mathcal{F}}_{\text{inst}}$, it is possible to instantiate the same vertex multiple times. Suppose, for a graph G, a uniqueness question $f_B \in \tilde{\mathcal{F}}_{\text{uniq}}$ receives a positive answer (i.e. $f_B(G) = 1$). Although the set B can be used to identify a vertex, say $v_1 \in V$, this identifying set may not be unique; in general there exists many $B \subset \Lambda_V$ that can instantiate the vertex v_1. Hence a vertex may be instantiated multiple times: it is possible that a uniqueness question $f_B \in \tilde{\mathcal{F}}_{\text{uniq}}$ receives a positive answer, yet

$$\Omega^{\text{Inst}}(G, F_k) = \Omega^{\text{Inst}}(G, F_{k+1}),$$

where F_{k+1} is the question sequence resulting from adding the question f_B to the sequence F_k. This is a problem because it is not known whether a newly instantiated vertex is the same as a previously instantiated vertex (at least without asking further questions). The instantiation questions in $\mathcal{F}_{\text{inst}}$, however, do not have this problem because they only operate over non-instantiated vertices. Hence, for any question $f_{B,F_k} \in \mathcal{F}_{\text{uniq}}$ that receives a positive answer, we have that:

$$\Omega^{\text{Inst}}(G, F_k) \subset \Omega^{\text{Inst}}(G, F_{k+1}),$$

where F_{k+1} is the question sequence resulting from adding the question f_{B,F_k} to the se-

quence F_k.

The second difference between these question spaces concerns their capacity to interpret the graph space. In general, the questions in $\mathcal{F}_{\text{uniq}}(\mathcal{B})$ allow the instantiation of vertices that could not have been instantiated by questions in $\tilde{\mathcal{F}}_{\text{uniq}}(\mathcal{B})$. Consider the collection of all subsets of Λ_V that allow a new instantiation (i.e. can instantiate a vertex that has not previously been instantiated) when using these question spaces:

$$\tilde{\mathcal{B}}_{\text{inst}}(G, F_k) = \{B \subset \Lambda_V \mid f_B(G) = 1 \text{ and a new node is instantiated}\}$$

$$\mathcal{B}_{\text{inst}}(G, F_k) = \{B \subset \Lambda_V \mid f_{B,F_k}(G) = 1 \text{ and a new node is instantiated}\},$$

where $f_B \in \tilde{\mathcal{F}}_{\text{uniq}}$ is a simple uniqueness question and $f_{B,F_k} \in \mathcal{F}_{\text{uniq}}$ is a complex uniqueness question. Then, we have that:

$$\tilde{\mathcal{B}}_{\text{inst}}(G, F_k) \subset \mathcal{B}_{\text{inst}}(G, F_k)$$

for all $G \in \mathcal{G}$ and all sequences F_k. Hence, the questions in $\mathcal{F}_{\text{inst}}$ provide a greater capacity for identifying vertices. This capacity comes at a cost however: the estimation of statistics for them tends to require more data. We consider estimation below.

4.5 Question Sequences

In the previous sections, we defined several types of questions for graphs; in this section, we consider the sequential selection of these questions. Loosely speaking, questions will be selected so as to maximize the expected information about unknown graphs. This selection criterion corresponds to the use of unpredictable questions; the answer to these questions cannot be reliably guessed, even given the answers to previous questions. We begin by describing unpredictable questions and the conditional probabilities of interest. After that, we relate unpredictable questions to the minimization of the expected entropy over a graph space. Finally, we consider their estimation.

4.5.1 Conditional Probabilities

Suppose we have some graph space $\mathcal{G}$ with a distribution P over it. Further, suppose we have an unknown graph $G \in \mathcal{G}$ that we are asking questions about. Let $F_k = (f_1, \ldots, f_k)$ be a sequence of questions; the answers to these questions on graph G are

$$\begin{aligned} A_k &= F_k(G) \\ &= (f_1(G), \ldots, f_k(G)). \end{aligned}$$

Define the conditional probability of a question f_{k+1} having a positive answer given a history $H_k = (F_k, A_k)$ as:

$$P_{H_k}(f_{k+1}(\mathbf{G}) = 1) \equiv P(f_{k+1}(\mathbf{G}) = 1 \mid F_k(\mathbf{G}) = A_k),$$

where $\mathbf{G}$ is a random graph.

4.5.2 Question Predictability

Suppose we have some history $H = (F, A)$, where $F = (f_1, \ldots, f_k)$ and $A = (a_1, \ldots, a_k)$ are sequences of questions and answers, respectively. Define the *predictability* of a question f with respect to history H as

$$\rho(f, H) = \left| P_H(f(\mathbf{G}) = 1) - 0.5 \right|.$$

A question f is *unpredictable* if

$$\rho(f, H) = 0.$$

We call these questions unpredictable because, without explicit knowledge of the graph, they cannot be reliably guessed, even given the answers to the previously asked questions. Let $\mathcal{F}$ denote some question space, and for a history H, let $\mathcal{U}_H$ denote the set of

ϵ-unpredictable questions in $\mathcal{F}$:

$$\mathcal{U}_H \equiv \{f \in \mathcal{F} \mid \rho(f, H) \leq \epsilon\}.$$

The question space $\mathcal{F}$ needs to be designed such that this set is never empty (i.e. $\mathcal{U}_H \neq \emptyset$ for some H), at least for question streams of a reasonable length. For example, even if our only goal is to instantiate vertices, the inclusion of existence questions in our question space is valuable; their use in sequences can often cause uniqueness questions that are normally predictable to become unpredictable.

4.5.3 Entropy Minimization

Questions that are unpredictable are desirable from an information-theoretic perspective: they minimize the expected entropy of a random graph. Let f be an unpredictable question with respect to history $H = (F, A)$, and define the event $H_{\mathcal{G}} \equiv \{G \in \mathcal{G} \mid F(G) = A\}$. Then, we have:

$$\begin{aligned} P_H(f(\mathbf{G}) = 1) = 0.5 \quad &\Leftrightarrow \quad \mathcal{H}(f(\mathbf{G}) \mid H_{\mathcal{G}}) = 1 \\ &\Rightarrow \quad f = \arg\min_{f' \in \mathcal{F}} \mathcal{H}(\mathbf{G} \mid H_{\mathcal{G}}, f'(\mathbf{G})) \end{aligned}$$

where $\mathcal{H}$ is the entropy, and where

$$\mathcal{H}(\mathbf{G} \mid H_{\mathcal{G}}, f(\mathbf{G})) = \sum_{a \in \{0,1\}} P(f(\mathbf{G}) = a \mid H_{\mathcal{G}})\, \mathcal{H}(\mathbf{G} \mid H_{\mathcal{G}}, f(\mathbf{G}) = a).$$

The above implication follows because

$$\mathcal{H}(\mathbf{G} \mid H_{\mathcal{G}}, f(\mathbf{G})) = \mathcal{H}(\mathbf{G}, f(\mathbf{G}) \mid H_{\mathcal{G}}) - \mathcal{H}(f(\mathbf{G}) \mid H_{\mathcal{G}})$$

and, since the question f is a function of a graph, we have

$$\mathcal{H}(\mathbf{G}, f(\mathbf{G}) \mid H_{\mathcal{G}}) = \mathcal{H}(\mathbf{G} \mid H_{\mathcal{G}}).$$

To summarize, we have shown that an unpredictable question minimizes the expected entropy of an unknown (random) graph.

4.6 Estimation

Let's consider statistical estimations. Suppose we have some graph space $\mathcal{G}$ with a distribution P over it, and suppose we have independently sampled a set of graphs from this distribution:

$$\mathbb{T}_n = \{G_1, G_2, \ldots, G_n\}.$$

We assume each graph in this dataset is known (e.g., they have been exhaustively annotated). Let $f : \mathcal{G} \to \mathbb{R}$ be some real function and suppose we want to estimate its expected value $\mu = \mathbb{E}(f(\mathbf{G}))$. This can be done using the empirical average:

$$\hat{\mu}(\mathbb{T}_n) = \frac{1}{n} \sum_{i=1}^{n} f(G_i). \tag{4.1}$$

Suppose we want to estimate the conditional expected value $\mu = \mathbb{E}(f(\mathbf{G}) \mid \mathbf{G} \in \mathcal{G}')$, where $\mathcal{G}' \subset \mathcal{G}$ is an event. This can also be done using the empirical average:

$$\hat{\mu}(\mathbb{T}_n) = \frac{\sum_{i=1}^{n} I_{\{G_i \in \mathcal{G}'\}} f(G_i)}{\sum_{i=1}^{n} I_{\{G_i \in \mathcal{G}'\}}}.$$

Now, suppose we have a question $f : \mathcal{G} \to \{0, 1\}$ and a history $H = (F, A)$ of questions and answers. The conditional probability $P_H(f(\mathbf{G}) = 1) = \mathbb{E}(f(\mathbf{G}) \mid F(\mathbf{G}) = A)$, and the estimate simplifies to the following form:

$$\hat{P}_H(f(\mathbf{G}) = 1) = \frac{\#\{G \in \mathbb{T} \mid F(G) = A, f(G) = 1\}}{\#\{G \in \mathbb{T} \mid F(G) = A\}}.$$

However, as the number of questions in the history increases, the data that coheres with the history will quickly become depleted, and hence estimates of the above form will degenerate. In chapter 5, we consider invariance assumptions to alleviate this problem.

4.7 Query Engine

Let $\mathcal{F}$ be some space of binary questions. We refer to a function that sequentially selects questions from $\mathcal{F}$ as a *query-engine*. To be more specific, a query-engine is a function Φ that takes as input a history H, a question space $\mathcal{F}$, and a set of training data $\mathbb{T} = \{G_1, G_2, \ldots, G_N\}$, and outputs a question $f \in \mathcal{F}$, the question to ask next in the sequence. By iteratively running a query-engine, we obtain a sequence of questions. The query-engine uses the training data to estimate an unpredictable question with respect to the history and has the form:

$$\Phi(H, \mathcal{F}, \mathbb{T}) = f$$

where $f \in \mathcal{F}$. Finally, to introduce randomness into question streams, a query-engine may output a question that is randomly selected from among the set of (estimated) unpredictable questions (i.e., a query engine can also correspond to a conditional distribution of the form $P(f \mid H, \mathcal{F}, \mathbb{T})$).

Chapter 5

A Visual Turing Test for Computer Vision Systems

In computer vision, as in other fields of AI, the methods of evaluation largely define the scientific effort. Most current evaluations measure detection accuracy, emphasizing the classification of regions according to objects from a pre-defined library. But detection is not the same as understanding. As an alternative, we construct a query engine, which is an operator-assisted device that produces a stochastic sequence of binary questions (a "visual Turing test") from a given test image. The test is easy to prepare, since it does not require a full semantic description of the test image: the engine proposes a question; the operator either provides the correct answer or rejects the question as ambiguous; the engine proposes the next question ("just-in-time truthing"). The test is then administered to the computer-vision system, one question at a time. After the system's answer is recorded, the system is

provided the correct answer and the next question. Parsing is trivial and deterministic; the system being tested requires no natural language processing. The query engine employs statistical constraints, learned from a training set, to produce questions with essentially unpredictable answers—the answer to a question, given the history of questions and their correct answers, is nearly equally likely to be positive or negative. In this sense, the test is only about vision. The system is designed to produce streams of questions that follow natural story lines, from the instantiation of a unique object, through an exploration of its properties, and onto its relationships with other uniquely instantiated objects.

5.1 Introduction

Going back at least to the mid-twentieth century there has been an active debate about the state of progress in artificial intelligence and how to measure it. Alan Turing [Turing, 1950] proposed that the ultimate test of whether or not a machine could "think," or think at least as well as a person, was for a human judge to be unable to tell which was which based on natural language conversations in an appropriately cloaked scenario. In a much-discussed variation (sometimes called the "standard interpretation"), the objective is to measure how well a computer can *imitate* a human [Saygin et al., 2003] in some circumscribed task normally associated with intelligent behavior, although the practical utility of "imitation" as a criterion for performance has also been questioned [Russell, 2003]. In fact, the overwhelming focus of the modern AI community has been to assess machine perfor-

mance more directly by dedicated tests for specific tasks rather than debating about general "thinking" or Turing-like competitions between people and machines.

In this chapter we implement a new, query-based test for computer vision, one of the most vibrant areas of modern AI research. Throughout this paper we use "computer vision" more or less synonymously with semantic image interpretation - "images to words." But of course computer vision encompasses a great many other activities; it includes the theory and practice of image formation ("sensors to images"); image processing ("images to images"); mathematical representations; video processing; metric scene reconstruction; and so forth. In fact, it may not be possible to interpret scenes at a semantic level without taking at least some of these areas into account, especially the geometric relationship between an image and the underlying 3D scene. But our focus is how to evaluate a system, not how to build one.

Beside successful commercial and industrial applications, such as face detectors in digital cameras and flaw detection in manufacturing, there has also been considerable progress in more generic tasks, such as detecting and localizing instances from multiple generic object classes in ordinary indoor and outdoor scenes, in "fine-grained" classification such as identifying plant and animal species, and in recognizing attributes of objects and activities of people. The results of challenges and competitions (see [Everingham et al., 2010], [Deng et al., 2009]) suggest that progress has been spurred by major advances in designing more computationally efficient and invariant image representations [Lowe, 2004], [Zhu et al., 2006], [Yu & Morel, 2009]; in stochastic and hierarchical modeling [Zhu & Mum-

ford, 2006], [Ommer et al., 2006], [Chang et al., 2010], [Lu et al., 2014]; in discovering latent structure by training multi-layer networks with large amounts of unsupervised data [Hinton et al., 2006]; and in parts-based statistical learning and modeling techniques [Li et al., 2003], [Amit & Trouve, 2007], [Felzenszwalb & Huttenlocher, 2005], especially combining discriminative part detectors with simple models of arrangements of parts [Felzenszwalb et al., 2009]. Quite recently, sharp improvements in detecting objects and related tasks have been made by training convolutional neural networks with very large amounts of annotated data [Krizhevsky et al., 2012], [Girshick et al., 2013], [Oquab et al., 2013], [Zhang et al., 2013], [Hariharan et al., 2014].

More generally, however, machines lag very far behind humans in "understanding images" in the sense of generating rich semantic annotation. For example, systems that attempt to deal with occlusion, context and unanticipated arrangements, all of which are easily handled by people, typically encounter problems. Consequently, there is no point in designing a "competition" between computer vision and human vision: interpreting real scenes (such as the ones in Figure 5.1) is virtually "trivial" (at least effortless and nearly instantaneous) for people whereas building a "description machine" that annotates raw image data remains a fundamental challenge.

We seek a quantitative measure of how well a computer vision system can interpret ordinary images of natural scenes. Whereas we focus on urban street scenes, our implementation could easily be extended to other image populations and the basic logic and motivations remain the same. The "score" of our test is based on the responses of a system

Figure 5.1: Urban street scenes. First row: Athens, Baltimore, Busan, Delhi. Second row: Hong Kong, Miami, Rome, Shanghai.

under evaluation to a series of binary questions about the existence of people and objects, their activities and attributes, and relationships among them, all relative to an image. We have chosen image-based rather than scene-based queries (see section 5.3).

Suppose an image sub-population $\mathcal{I}$ has been specified ("urban street scenes" in Figure 5.1), together with a "vocabulary" and a corresponding set of binary questions. Our prototype "visual Turing test" (VTT) is illustrated in Figure 5.2 (and additional examples are shown in appendix A). Questions are posed sequentially to the computer vision system using a "query language" which is defined in terms of an allowable set of predicates. The interpretation of the questions is unambiguous and does not require any natural language processing. The core of the VTT is an automatic "query generator" which is learned from annotated images and produces a sequence of binary questions for any given "test" image $I_0 \in \mathcal{I}$ whose answers are "unpredictable". In loose terms, this means that hearing the first $k-1$ questions and their true answers for I_0 without actually seeing I_0 provides no information about the likely answer to the next question. In order to prepare for the test,

designers of the vision systems would be provided with the database used to train the query generator as well as the full vocabulary and set of possible questions, and would have to provide an interface for answering questions. One simple measure of performance is the average number of correct responses over multiple runs with different test images.

5.2 Current Evaluation Practice

Numerous datasets have been created to benchmark performance, each designed to assess some vision task (e.g., object detection) on some image domain (e.g., street-scenes). Systems are evaluated by comparing their output on these data to "ground-truth" provided by humans. One well-studied task is classifying an entire image by a general category, either at the object level ("car," "bike," "horse," etc.), where popular annotated datasets include the relatively small *Caltech-256* dataset [Griffin et al., 2007] with 256 classes and the far larger *ImageNet* dataset [Deng et al., 2009], or at the scene-level ("beach," "kitchen," "forest," etc.); see for example the SUN dataset [Xiao et al., 2010]. A natural extension of object-level image categorization is detecting and localizing all instances from generic classes in complex scenes containing multiple instances and events; localization refers to providing either a "bounding box" per instance or segmenting the object from the background. Popular datasets for this task include the *Pascal* dataset [Everingham et al., 2010], the *LabelMe* dataset [Russell et al., 2008], and the *Lotus Hill* dataset [Yao et al., 2007], all populated by relatively unconstrained natural images, but varying considerably in size

and in the level of annotation, ranging from a few keywords to hierarchical representations (*Lotus Hill*). Finally, a few other datasets have been assembled and annotated to evaluate the quality of detected object attributes such as color, orientation and activity; examples are the *Core* dataset [Endres et al., 2010], with annotated object parts and attributes, and the *Virat* dataset [Oh et al., 2011] for event detection in videos.

Why not continue to measure progress in more or less the same way with common datasets dedicated to sub-tasks, but using a richer vocabulary? First, as computer vision becomes more ambitious and aims at richer interpretations, it would seem sensible to fold these sub-tasks into a larger endeavor; a system which detects activities and relationships must necessarily solve basic sub-tasks anyway. Then why not simply require competing systems to submit much richer annotation for a set of test images than in previous competitions and then rank systems according to consistency with ground truth supplied by human annotators? The reason, and the justification for the VTT, is that the current method does not scale with respect to the richness of the representation. Even for the sub-tasks in the competitions mentioned earlier, the evaluation of performance, i.e., comparing the output of the system (e.g., estimated bounding boxes) to the ground-truth is not always straightforward and the quality of matches must be assessed [Zdemir et al., 2010]. Moreover, annotating every image submitted for testing at massive levels of detail is not feasible. Hence, objectively scoring the veracity of annotations is not straightforward. As in school, answering specific questions is usually more objective and efficient in measuring "understanding." Finally, some selection procedure seems unavoidable; indeed, the num-

ber of possible binary questions which are both probing and meaningful is virtually infinite. However, selecting a subset of questions (i.e., preparing a test) is not straightforward. We would argue that the only way to ask very detailed questions without having their answers be almost certainly "no" is sequential and adaptive querying—questions which build on each other to uncover semantic structure. In summary, the VTT is one way to "scale up" evaluation.

5.3 Proposed Test: Overview

In this section, we continue to summarize the key aspects of the VTT and challenges in constructing it in a non-technical fashion. Additional details can be found in [Geman et. al, 2015] and in its supplemental information.

Scenes vs. images. Our system is image-centered. We pose our questions succinctly in the form *"Is there a red car?"* where this is understood to mean *"Is there an instance of a red car in the scene partially visible in the image?"*. Similarly, given a designated rectangle of image pixels (see Figure 5.2 for some examples), the query *"Is there a person in the designated region?"* is understood to mean *"Is there an instance of a person in the scene partially visible in the designated image region?"*. The motivation for this choice of formatting questions is that questions that reference only the scene are too ambiguous for our purposes. What is the scope of "the scene"? The universal qualifier "partially visible

in the image" (or in the designated region) avoids this issue and leads naturally to instantiation and story lines. Finally, there is no essential difference between "partially visible in the image" and "partially visible in the designated region," making for an efficient way to determine unique instances.

Estimating uncertainty. The justification for counting all questions the same is the property of unpredictability: at each step k, the likelihood that the true answer for question k is "yes" given the true answers to the previous $k-1$ questions is approximately one-half. However, generating long strings of "interesting" questions and "story lines" is not straightforward due to "data-fragmentation": a purely empirical solution based entirely on collecting relative frequencies from an annotated training subset of size n from $\mathcal{I}$ is only feasible if the number of questions posed is approximately $\log_2 n$. Our proposed solution is presented as part of the Statistical Formulation; it rests on enlarging the number of images in the dataset which satisfy a given history by making carefully chosen invariance and independence assumptions about objects and their attributes and relationships.

Human in the loop. The operator serves two crucial functions: removing ambiguous questions and providing correct answers. Given a rich family of questions, some will surely be ambiguous for any specific test image. The solution is "just-in-time truthing": any question posed by the query generator can be rejected by the operator, in which case the generator supplies another nearly unpredictable one, of which there are generally many. The correct

1. Q: Is there a person in the blue region? **A: yes**
2. Q: Is there a unique person in the blue region? **A: yes**
(Label this person 1)
3. Q: Is person 1 carrying something? **A: yes**
4. Q: Is person 1 female? **A: yes**
5. Q: Is person 1 walking on a sidewalk? **A: yes**
6. Q: Is person 1 interacting with any other object? **A: no**
⋮
9. Q: Is there a unique vehicle in the yellow region? **A: yes**
(Label this vehicle 1)
10. Q: Is vehicle 1 light-colored? **A: yes**
11. Q: Is vehicle 1 moving? **A: no**
12. Q: Is vehicle 1 parked and a car? **A: yes**
⋮
14. Q: Does vehicle 1 have exactly one visible tire? **A: no**
15. Q: Is vehicle 1 interacting with any other object? **A: no**
17. Q: Is there a unique person in the red region? **A: no**
18. Q: Is there a unique person that is female in the red region? **A: no**
19. Q: Is there a person that is standing still in the red region? **A: yes**
20. Q: Is there a unique person standing still in the red region? **A: yes**
(Label this person 2)
⋮
23. Q: Is person 2 interacting with any other object? **A: yes**
24. Q: Is person 1 taller than person 2? **A: amb.**
25. Q: Is person 1 closer (to the camera) than person 2? **A: no**
26. Q: Is there a person in the red region? **A: yes**
27. Q: Is there a unique person in the red region? **A: yes**
(Label this person 3)
⋮
36. Q: Is there an interaction between person 2 and person 3? **A: yes**
37. Q: Are person 2 and person 3 talking? **A: yes**

Figure 5.2: A selection of questions extracted from a much longer sequence. Answers, including identifying Q24 as ambiguous, are provided by the operator (see paragraph on "Human in the loop"). Localizing questions include, implicitly, the qualifier "partially visible in the designated region" and instantiation (existence and uniqueness) questions implicitly include "not previously instantiated." The localizing windows used for each of the four instantiations (vehicle 1, person 1, person 2, and person 3) are indicated by the colored rectangles (blue—thick border, red—thin border, yellow—broken border). The colors are included in the questions for illustration. In the actual test, each question designates a single rectangle through its coordinates, so that "Is there a unique person in the blue region" would actually read "Is there a unique person in the designated region."

answers may or may not be provided to the system under evaluation at run time. Needless to say, given the state of progress in computer vision, neither of these roles can be served by an automated system. The test can be constructed either offline or "online" (during the evaluation). In either case, the VTT is "written" rather than "oral" since the choice of questions does not depend on the responses from the system under evaluation.

Instantiation. As discussed in the previous chapter, a key mechanism for arriving at semantically interesting questions is instance "instantiation." A series of positive answers to inquiries about attributes of an object will often imply a single instance, which can then be labeled as "instance k". Hence, questions which explicitly address uniqueness are also included, which usually become viable, that is close to unpredictable, after one or two attributes have been established. Once this happens, there is no ambiguity in asking whether "person 1" and "person 2" are *talking* or whether "person 1" is *occluding* "vehicle 2"; see Figure 5.2. We regard instantiation as identifying the "players" in the scene, allowing for story lines to develop.

Evolving descriptions. The statistical constraints naturally impose a "coarse-to-fine" flow of information, from gist to semantic detail. Due to the unpredictability criterion, the early questions can only inquire about coarse scene properties, such as *"Is there a person in the lefthand side of the image?"* or *"Is there a person wearing a hat?"*, because only these have intermediate probabilities of occurrence in the general population. It is only

after objects have been instantiated, i.e., specific instances identified, that the likelihoods of specific relationships among these "players" become appreciably greater than zero.

5.4 Vocabulary and Questions

Vocabulary. Our vocabulary $\mathcal{V}$ consists of three components: *types* of objects, $\mathcal{T}$, type-dependent *attributes* of objects, $\{\mathcal{A}_t, t \in \mathcal{T}\}$, and type-dependent *relationships* between two objects, $\{\mathcal{R}_{t,t'}\}$. For example, for "urban street scenes," some natural types (or categories) are people, vehicles, buildings, and "parts" such as windows and doors of cars and buildings. Attributes refer to object properties such as clothing and activities of people, or types and colors of vehicles. There may also be attributes based on localizing an object instance within an image, and these provide an efficient method of instantiation (see below). Relationships between two types can be either "ordered," for instance a person *entering* a car or building, or "unordered," for instance two people *walking* or *talking* together. And some relationship questions may depend on the position of the camera in the underlying 3D scene, such as asking which person or vehicle is closer to the camera.

Questions. Each question $f \in \mathcal{F}$ belongs to one of four categories: existence questions, $\mathcal{F}_{\text{exist}}$, uniqueness questions, $\mathcal{F}_{\text{uniq}}$, attribute questions, $\mathcal{F}_{\text{att}}$, or relationship questions, $\mathcal{F}_{\text{rel}}$. The goal of the existence and uniqueness questions is to instantiate objects, which are then labeled ("person 1," "vehicle 3," ...) and subsequently available, by reference to the

label, in attribute and relationship questions ("Is person 1 partially occluding vehicle 3?"). Consequently, questions in $\mathcal{F}_{\text{att}}$ and $\mathcal{F}_{\text{rel}}$ refer only to previously instantiated objects. See Figure 5.2 for examples drawn from $\mathcal{F}_{\text{exist}}$ (e.g., 1, 19, 26), $\mathcal{F}_{\text{uniq}}$ (e.g., 2, 9, 17), $\mathcal{F}_{\text{att}}$ (e.g., 3, 10, 23), and $\mathcal{F}_{\text{rel}}$ (e.g., 25, 36, 37).

As already mentioned, we use "in the designated region" as shorthand for "in the scene that is partially visible in the designated region of the image." Similarly, so as to avoid repeated discovery of the same objects, all existence and uniqueness questions include the additional qualifier "not previously instantiated," which is always implied rather than explicit. So "Is there a person in the designated region wearing a hat?" actually means "Is there a person in the scene partially visible in the designated region of the image, wearing a hat and not previously instantiated?"

Summarizing, the full set of questions is

$$\mathcal{F} = \mathcal{F}_{\text{exist}} \cup \mathcal{F}_{\text{uniq}} \cup \mathcal{F}_{\text{att}} \cup \mathcal{F}_{\text{rel}}$$

We assume the answers are unambiguous *for humans* in nearly all cases. However, there is no need to identify *all* ambiguous questions for any image. Filtering is "as needed": given $I_0 \in \mathcal{I}$, any question f which is elicited by the query generator but is in fact ambiguous *for* I_0 will be rejected by the human operator during the construction of the VTT. (An example is question 24 in the partial stream shown in Figure 5.2.)

5.5 Statistical Formulation

Selecting questions whose answers are unpredictable is only meaningful in a statistical framework in which answers are random variables relative to an image population $\mathcal{I}$, which serves as the underlying sample space, together with a probability distribution P on $\mathcal{I}$.

Query generator. Given an image $I \in \mathcal{I}$, the query generator interacts with an oracle (human being) to produce a sequence of questions and correct answers. The human either rejects a question as ambiguous or provides an answer, in which case the answer is assumed to be a (deterministic) function of I, i.e., every question is a function of the form $f : \mathcal{I} \to \{0, 1\}$. The process is recursive: given a *history* of binary questions and their answers, $H = \big((f_1, a_1), \ldots, (f_k, a_k)\big)$, $f_i \in \mathcal{F}$ and $a_i \in \{0, 1\}$, the query generator either stops, for lack of additional unpredictable questions, or proposes a next question f, which is either rejected as ambiguous or added to the history along with its correct answer a:

$$H \to [H, (f, a)] \triangleq \big((f_1, a_1), \ldots (f_k, a_k), (f, a)\big), \;\; a \in \{0, 1\}$$

Not all sequences of questions and answers make sense. In particular, attribute and relationship questions ($\mathcal{F}_{att}$ and $\mathcal{F}_{rel}$) always refer to previously instantiated objects, restricting the set of meaningful histories, which we shall denote by $\mathbb{H}$. A key property of histories $H = \big((f_1, a_1), \ldots (f_k, a_k)\big) \in \mathbb{H}$ produced by the query generator is that each question f_i, given the history $\big((f_1, a_1), \ldots (f_{i-1}, a_{i-1})\big)$, is "unpredictable," a concept which we will

now make precise.

Given a history H, only some of the questions $f \in \mathcal{F}$ are good candidates for followup. As already noted, references to labeled objects cannot precede the corresponding instantiation questions, and furthermore there is a general ordering to the questions designed to promote natural story lines. For a given query generator, we will write $\mathcal{F}_H$ to indicate the set of possible followup questions defined by these *non-statistical* constraints. Typically, $\mathcal{F}_H$ contains many candidates, most of which are highly predictable given the history H, and therefore unsuitable.

In the previous chapter, we discussed the empirical estimation of the conditional probability of a question having a positive answer, given a history; we repeat this here for convenience. The set of histories, $\mathbb{H}$, can be viewed as a set of binary random variables: $H = H(I) = 1$ if $H = \big((f_1, a_1), \ldots (f_k, a_k)\big) \in \mathbb{H}$ and if the sequence of questions $(f_1, \ldots, f_k)$ produces the sequence of unambiguous answers $(a_1, \ldots, a_k)$ for the image I, and $H = 0$ otherwise. We will write P_H for the *conditional* probability on $\mathcal{I}$ given that $H(I) = 1$.

Consider now the probability under P_H that a question $f \in \mathcal{F}_H$ elicits the (unambiguous) response $a \in \{0, 1\}$, for a given history $H \in \mathbb{H}$:

$$P_H(f(\mathbf{I}) = a) \triangleq \frac{P\{I : H(I) = 1, f(I) = a\}}{P\{I : H(I) = 1\}}, \tag{5.1}$$

where $\mathbf{I}$ is a random image. Under P_H, $f(\mathbf{I})$ is a binary random variable which may or may

not be "unpredictable." To make this precise, we define the *predictability* of $f \in \mathcal{F}_H$, given the history $H \in \mathbb{H}$, by $\rho_H(f) = |P_H(f(\mathbf{I}) = 1) - 0.5|$. Evidently, $\rho = 0$ indicates f is totally unpredictable and $\rho = 0.5$ indicates f is totally predictable.

Randomization. In general, many questions have answers with low predictability at each step k. Rather than select the *most* unpredictable question at step k, we make a random selection from the set of *almost unpredictable* questions, defined as those for which $\rho_H(q) \leq \epsilon$, where H is the history preceding the k'th question. (In practice we choose $\epsilon = 0.15$, and we designate all such questions "unpredictable.") In this way, we can generate many query streams for a given test image I, and develop multiple story lines within a query stream. In doing so, a path to instantiation might be the following sequence of questions and answers:

1. Q: "*Is there an object of type t with attribute a?*" A: "*yes*"
2. Q: "*Is there an object of type t with attribute b?*" A: "*yes*"
3. Q: "*Is there a unique object of type t with attributes a and b?*" A: "*yes*"

Once it is established that there is an object with attribute a and an object with attribute b, then the likelihood of having a unique instance with *both* attributes may rise to approximately one-half. Commonly, a designated region serves as an important instantiating attribute, as in the sequence:

1. Q: "*Is there an object of type t in region w?*" A: "*yes*"

2. Q: "*Is there a unique object of type t in region w?*" A: "*no*"

3. Q: "*Is there an object of type t in region w with attribute a?*" A: "*yes*"

4. Q: "*Is there a unique object of type t in region w with attribute a?*" A: "*yes*"

Here, for example, t might refer to a person, of which several are partially visible in region w, but only one possesses the additional attribute a (e.g., "sitting", "female", or "wearing a hat"). There are more examples in Figure 5.2.

Story lines and the simplicity preference. We impose constraints on the set of questions allowed at each step—the set of available followup questions given the history H, which we have denoted by $\mathcal{F}_H$, is a small subset of the set of all possible questions, $\mathcal{F}$. The main purpose is to encourage natural sequences, but these constraints also serve to limit the number of conditional likelihoods that must be estimated.

The loop structure of the query engine enforces a general question flow that begins with existence and uniqueness questions ($\mathcal{F}_{\text{exist}}$, $\mathcal{F}_{\text{uniq}}$), with the goal of instantiating objects. As objects are instantiated, the vision system is interrogated about their properties, meaning their attributes, and then their relationships to the already-instantiated objects. After these "story lines" are exhausted, the outer loops are revisited in search of new instantiations. The query engine halts when there are no more unpredictable existence or uniqueness questions. As already mentioned, all loops include randomization, meaning that the next query is randomly selected from the questions in $\mathcal{F}_H$ that are found to be unpredictable.

The pose attribute is especially useful to an efficient search for uniquely characterized objects, i.e. instantiation. Once the existence of an object that is partially visible within a region w is established, ensuing existence and uniqueness queries are restricted to w or its sub-regions. As these regions are explored, the unpredictability constraint then favors questions about the same object type, but annotated with additional attributes. Eventually, either an object partially visible in a sub-region of w is instantiated or the collection of unpredictable questions about such an object is exhausted. In the latter case the query engine returns to the outer loop and begins a new line of questions; in the former, it explores the attributes and relationships of the newly instantiated object. (All regions are rectangular.)

Finally, there is a simplicity constraint that further promotes a natural line of questions. This can be summarized, roughly, as "one new thing at a time." An existence, uniqueness, or attribute question, f, is considered simpler than an alternative question of the same type, f', if f contains fewer attributes than f'. Given the unpredictable subset of $\mathcal{F}_H$, simpler questions are favored over more complex questions, and questions of equal complexity are chosen from with equal likelihood.

Estimating predictability. The conditional likelihoods, $P_H(f(\mathbf{I}) = 1)$, are estimated from a training set $\mathbb{T}$ in which all answers (or equivalent information—see Figure 5.3) are provided for each of n images from $\mathcal{I}$. The methods used to gather and annotate the training images are discussed in the next section, on the prototype VTT. The objects, people and vehicles, are located with bounding boxes and labelled with their attributes, and pairs of

objects are labelled with their relationships.

The task of estimating conditional likelihoods, and therefore predictability, is guided in part by the ordering of questions built into the query engine, which, as already noted, begins with a search for an instantiated object, immediately followed by questions to determine its attributes, and then finally by an exploration of its relationships with any previously instantiated objects.

For instantiation questions, $f \in \mathcal{F}_{inst} \triangleq \mathcal{F}_{exist} \cup \mathcal{F}_{uniq}$, the natural estimator $\hat{P}_H(f(\mathbf{I}) = 1)$ is the relative frequency (maximum likelihood) estimator

$$\frac{\#\{I \in \mathbb{T} : H(I) = 1, f(I) = 1\}}{\#\{I \in \mathbb{T} : H(I) = 1\}} \tag{5.2}$$

Observe, though, that the number of images in the training set which satisfy the history H (i.e., for which $H(I) = 1$) is cut approximately in half at each step, and hence after about $\log_2 n$ steps direct estimation is no longer possible. Consequently, to generate tests with more than ten or so questions, we are obliged to make "invariance" assumptions to allow for data pooling so as to expand the number of images from which these relative frequencies are computed. Specifically, if we assume that the random variable $f(\mathbf{I})$, $f \in \mathcal{F}_{inst}$, given the history $H \in \mathbb{H}$, depends only on a subsequence, H'_f of H, then the distribution on $f(\mathbf{I})$ is invariant to the questions and answers in H that were dropped, and the estimator **[5.2]** can be modified by substituting the condition $H(I) = 1$ by $H'_f(I) = 1$.

Let $w \in \mathcal{W}$ be the localizing region, possibly the entire image, referenced in the in-

stantiation question f. H'_f is derived from H by assuming that the event $f(\mathbf{I}) = a$ is independent of all attribute and relationship questions in H, and all existence and uniqueness questions which involve localizations $w' \in \mathcal{W}$ which are disjoint from w, with the important exception of uniqueness questions that answered positive ($f' \in \mathcal{F}_{\text{uniq}}$, $f'(\mathbf{I}) = 1$) and therefore instantiated a new object. In other words, the approximation is that, conditioned on the history, the distribution of an instantiation question depends only on the uniqueness questions that instantiated objects, and the existence and uniqueness questions that are localized to regions intersecting w. By preserving the instantiating questions in H, which addresses the potential complications introduced by the implied qualifier "not previously instantiated," we guarantee that $H(I) = 1 \Rightarrow H'_f(I) = 1$ for all $I \in \mathbb{T}$, so that the population of images used to estimate $P_H(f(\mathbf{I}) = 1)$ with $H'_f(I)$ is no smaller than the one with $H(I)$ and typically far larger.

As for attribute questions, $f \in \mathcal{F}_{\text{att}}$, which are always about the most recently instantiated object and always precede any relational information, the natural (relative frequency) estimator for $P_H(f(\mathbf{I}) = 1)$ is in terms of the population of labelled objects found in the training images, rather than the images themselves. Given a history H, consider a question of the form "Does object o_t have attribute a?", where o_t is an object of type $t \in \{$*person, vehicle*$\}$ and $a \in \mathcal{A}_t$. The history, H, defines a (possibly empty) set of attributes, denoted A, that are already known to belong to o_t. Let $\mathcal{O}_{\mathbb{T}}$ be the set of all annotated objects in the training set, and, for each $o \in \mathcal{O}_{\mathbb{T}}$, let $\mathcal{T}_{\mathbb{T}}(o)$ be the type of o and $\mathcal{A}_{\mathbb{T}}(o)$ be the set of attributes belonging to o, e.g., $\mathcal{T}_{\mathbb{T}}(o) =\{$*person*$\}$ and $\mathcal{A}_{\mathbb{T}}(o) =\{$*female,*

adult, standing$\}$ for the right-most object in Figure 5.3. The relative frequency estimator for $P_H(f(\mathbf{I}) = 1)$, using the population of annotated objects, is

$$\frac{\#\{o \in \mathcal{O}_{\mathbb{T}} : \mathcal{T}_{\mathbb{T}}(o) = t, A \cup \{a\} \subseteq \mathcal{A}_{\mathbb{T}}(o)\}}{\#\{o \in \mathcal{O}_{\mathbb{T}} : \mathcal{T}_{\mathbb{T}}(o) = t, A \subseteq \mathcal{A}_{\mathbb{T}}(o)\}} \tag{5.3}$$

There is again the sparsity problem, which we address in the same way—through invariance assumptions that effectively increase the number of objects. The set of attributes for objects of type t can be partitioned into subsets that can be reasonably approximated as independent conditioned on belonging to a particular object o_t. As an example, if $t =$*person* then *crossing a street* is not independent of *standing still*, but both are approximately independent of gender, $\{$*male, female*$\}$, and of *child* versus *adult*, as well as whether or not o_t is *carrying something* or *wearing a hat*. These conditional independence assumptions decrease the size of the set A in **[5.3]**, thereby increasing the set of $o \in \mathcal{O}_{\mathbb{T}}$ used to estimate $P_H(f(\mathbf{I}) = 1)$.

The approach to relationship questions, $f \in \mathcal{F}_{\text{rel}}$, is essentially the same as the approach to attribute questions, except that the training population is the set of *pairs* of objects in the training images, rather than the individual objects. The independence (invariance) assumptions include relationships that are independent of the attributes of the related objects (e.g., the relationship *driving/riding* a vehicle is assumed to be independent of both the gender of the person driving or riding, as well as whether the vehicle is dark or light colored, or whether or not its tires are visible) and relationships that are independent of each other

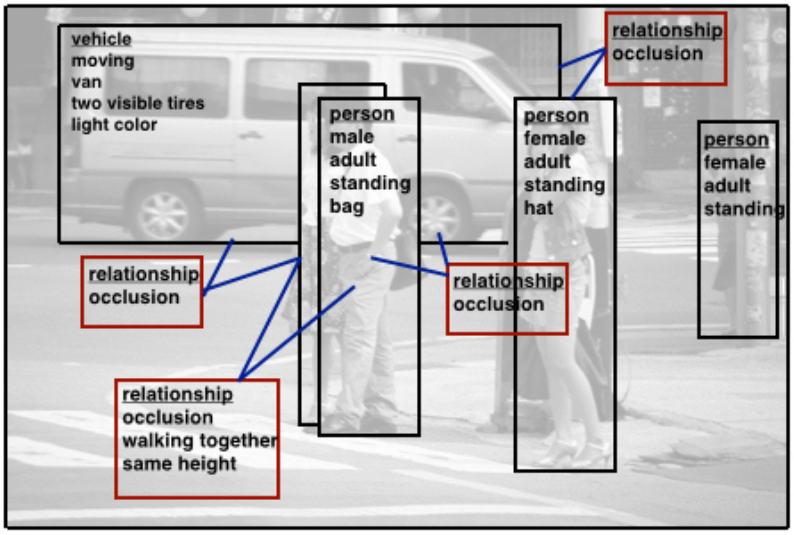


Figure 5.3: Annotation provided by human workers.

(e.g., whether one vehicle is closer to the camera than another vehicle is assumed to be independent of which vehicle is larger).

5.6 A Prototype VTT

The data collection and annotation was performed by undergraduate workers at Johns Hopkins University. Unlike "crowd-sourcing", this allowed for more customized instructions. Our dataset has $2{,}591$ images, collected online using search engines such as Google street view and required to meet certain basic criteria: portray a standard city street scene; be obtained during daytime; have a camera height from roughly head-level to several feet above; contain clearly visible objects, attributes, and relationships from our vocabulary. The images are from large cities from many countries.

For annotation, we can rule out directly answering each binary question $f \in \mathcal{F}$, since

the questions only make sense in the context of a history—$\mathcal{F}_{\text{att}}$ and $\mathcal{F}_{\text{rel}}$ always refer to instantiated objects, and $\mathcal{F}_{\text{exist}}$ and $\mathcal{F}_{\text{uniq}}$ always include the not-previously-instantiated qualification. As discussed, a history itself can be viewed as a binary function of the image, but there are far too many for an exhaustive annotation. Instead, an essentially equivalent, but more compact and less redundant, representation was used. For example, once a "bounding box" is provided for every object (see the examples in Figure 5.3), there is a high likelihood that the correct answer to a localization question "Is x partially visible in region w?" is determined by whether or not the bounding box of x intersects the region w. Thus bounding boxes were drawn around every instance of an object for which the annotator had no uncertainty about its category. For partially occluded objects, the bounding box was placed over the region of the image that the annotator expected the object would occupy had the object not been partially occluded. Attributes were annotated only for objects in which all the attributes were unambiguous, which alleviated the annotation of distant objects. Relationships were only annotated between pairs of objects with bounding boxes and for which at least one relationship from the type-dependent list was present.

The prototype includes only two types of objects: $\mathcal{T} = \{$*people, vehicles*$\}$. However, we also consider a few "parts"—things carried by people, and tires of vehicles, folded into the attribute categories. There is also an "attribute" for every element of a multi-scale collection $\mathcal{W}$ of rectangular subsets of pixels referred to as "regions" ; the "attribute" corresponding to $w \in \mathcal{W}$ is that the object instance is partially visible within w. The set of attributes also includes properties which are independent of positioning in the underlying

scene, such as *female, child, wearing a hat, carrying something* for type *people*, and *car, truck, motorcycle, bicycle, light colored* for type *vehicle*. Still others refer to pose and context—e.g., *sitting, crossing a street, walking on a sidewalk, entering/exiting a building* for *people*, and *moving, stopped, parked, one tire visible, two tires visible* for *vehicles*. Additionally, for both people and vehicles, the attribute *interacting with something* refers to any of a specific collection of relationships: for a person, *talking, walking together, holding hands* with another person, or *driving/riding, exiting, entering* a vehicle, and for a vehicle, *immediately behind, immediately in front* of another vehicle.

Relationships could be ordered or unordered. The unordered relationships between people are *talking, walking together, holding hands* and the ordered ones are about which person is *taller, closer to the camera*, and possibly *occluding* the other. The ordered relationships for two vehicles are the same, with *taller* replaced by *larger* and the addition of *immediately behind, immediately in front*; there are no unordered relationships between vehicles. Finally, a person may be *driving/riding*, *exiting*, *entering*, *occluding*, or *occluded by* a vehicle.

Level of difficulty. The vocabulary was selected to avoid query streams that would be considered hopelessly difficult by today's computer-vision standards. Nevertheless, there are plenty of subtleties to challenge, and likely defeat, the best existing systems. A few additions to the vocabulary would dial up the difficulty, considerably, say adding the relationship "playing catch" or other objects like windows, signs, and tables and chairs, which

are often nearly impossible to identify without context, especially when partially occluded.

5.7 Discussion

In the decades following Alan Turing computer vision became one of the most active areas of AI. The challenge of making computers "see" has attracted researchers from across science and engineering and resulted in a highly diverse set of proposals for formulating the "vision problem" in mathematical terms, each with its ardent advocates. The varying popularity of competing strategies can be traced in the proceedings of conferences.

Debates persist about what actually works and how to measure success. Until fairly recently, each new method was "validated" on homegrown data and with homegrown metrics. Recently, the computer vision community has accepted testing on large common datasets, as reviewed above, and various well-organized "challenges" have been accepted by many research groups. Many believe that adopting uniform metrics has made it easier to sort out what works appreciably better than before and accelerated progress.

But these metrics, such as false positive and false negative rates for sub-tasks such as detecting and localizing people, do not yet apply to the richer descriptions that human beings can provide, for example in applying contextual reasoning to decide whether or not a car is "parked" or is "larger" than another, or a person is "leaving" a building or "observing" something, or two people are "walking and talking together." If annotating ordinary scenes with such precision is accepted as a benchmark for vision, then we have argued for raising

the bar and proceeding directly to metrics for full-scale scene interpretation. We have proposed a "written" VTT as a step in this direction.

Many design decisions were made, some more compelling than others. "Story lines" approximate natural sequences of questions and are well handled by the loop structure of the algorithm. On the other hand, whereas conditional independence assumptions are probably a necessary approach to the data sparsity problem, the prototype lacks a unified implementation. Scaling to substantially larger vocabularies and more complex relationships, and deeper part/whole hierarchies, would be difficult to manage by simply enlarging the existing brute-force tabulation of dependency relationships. Possibly, the right approach is to build full-blown generative scene models, at least for the placements of parts and objects, and object groupings, from which predictability could be estimated via sampling or inferred by direct calculation.

Finally, coming back to a "conversation" with a machine, another possibility is a more free-form, open-ended "oral test": the operator formulates and delivers a query to the system under evaluation, awaits an answer, and then chooses the next query, presumably based on the history of queries and system answers. As before, the operator may or may not provide the correct answer. This has the advantage that the operator can "probe" the system capacities with the singular efficiency of a human, for example detect and focus on liabilities and ask "confirmatory" questions. But the oral test has the disadvantage of being subjective and requiring rapid, basically real-time, responses from the system. On balance, the written test seems to be more practical, at least for the time being.

5.7.1 Issues

For completeness, we mention some of the weak points of the proposed visual Turing test, at least from a practitioner's perspective. (This section contains personal opinions that do not necessarily reflect those of more senior researchers involved in this work.) One issue is that the test, in essence, requires a large amount of annotated data in order to decide what data to annotate; this belies the basic assumption about exhaustive annotation being infeasible, and moreover, if this annotated data were available, then it could be directly used for evaluation purposes anyways.

Another issue concerns the tests use of binary questions; no matter how well they are chosen, answering binary questions about an image is inefficient, in terms of effort, for human annotators. Consider, for example, how a pixel in an image generally belongs to at most one object, and this object is only one out of a vast number of possible objects. Humans are naturally able to both handle and exploit this sparsity in imagery. As a result, it is faster for a person to specify what is in an image, than everything that is not in an image. Using our natural ability to handle sparsity becomes increasingly important as the representation space becomes increasingly complex, and this has to be done by using non-binary questions, i.e., questions with a larger answer space.

Finally, we mention that the word Turing is used because of the test's use of questions, which somewhat resembles the original Turing test. Technically, however, almost any test used in the computer vision literature is equivalent to asking questions, as this is rather general.

Chapter 6

Scene Modeling

In this chapter, we develop a statistical model for real-world scenes. A scene is composed of objects (e.g., people and vehicles), where each object has attributes (e.g., wearing a hat, parked), and there are relationships between objects (e.g., holding hands, talking). Due to the complexity of scenes, we develop the model in steps. We begin by modeling the spatial configuration of objects in an image, representing them with rectangles. We refer to this random set problem as *random rectangles*. Next, we expand the model to a random graph model by incorporating edges and attributes.

The motivation for developing this scene model is as follows. In the previous chapter, we presented a visual Turing test that asks unpredictable questions about imagery, and to estimate predictability, invariance assumptions were used. The problem, however, is that these estimates lack rigor: it is possible to estimate a set of conditional probabilities that are *inconsistent* in the sense that there does not exist a probability distribution that

could have produced it. Suppose we have a finite graph space $\mathcal{G}$ with a distribution over it. Let $\mathcal{P}$ denote the collection of conditional probabilities corresponding to histories of questions and answers, and let $\hat{\mathcal{P}}$ denote a collection of estimates of these conditional probabilities. It is well-known that if the set $\hat{\mathcal{P}}$ is obtained using empirical estimates, then it is consistent. However, if $\hat{\mathcal{P}}$ is obtained with invariance assumptions in the estimates, as in the previous chapter, then this result is no longer true. To be more specific, if these assumptions are applied to each estimate individually, not accounting for their effect on the set as a whole, then inconsistencies can result. Hence, to ensure consistency, we will incorporate invariance assumptions into a statistical model here.

After learning a distribution from this model, we use it to produce unpredictable question sequences. We will discuss the problem of conditional sampling; in general, this sampling is difficult because the evidence resulting from question sequences is complex. However, under certain assumptions on the question space, the conditional sampling becomes amenable to computation.

6.1 Scene Graphs

We represent a scene as a graph in which vertices represent objects and edges represent relationships between objects. Recall, in the previous chapter, objects were assigned a type $t \in \mathcal{T}$, a set of type-dependent attributes $A \subset \mathcal{A}_t$, and a bounding box $r \in \mathcal{R}$. We use the same set of objects here; let a vertex have the form $v = (t, A, r)$ and let the vertex space

Λ_V be:

$$\Lambda_V = \{(t, A, r) \mid t \in \mathcal{T}, A \subset \mathcal{A}_t, r \in \mathcal{R}\}.$$

For simplicity, we restrict the set $\mathcal{R}$ to rectangles with a center point on a grid $\{1, \ldots, N\}^2$, having either a vertical or horizontal orientation, and taking a size of either small, medium, or large. Also for simplicity, we let the edge space be $\Lambda_E = \{0, 1\}$, representing the presence or absence of an interaction[1] between two objects. Thus, we represent a scene as a graph $G = (V, E)$, where

$$V \subset \Lambda_V$$

$$E : V \times V \to \Lambda_E.$$

Let $\mathcal{G}$ be the space of all such graphs.

6.2 Random Rectangles

Before developing a distribution over the graph space $\mathcal{G}$, due to the complexity of the vertices, we try developing a distribution over them. In particular, we focus on modeling

[1]The definition of an interaction between two objects depends on the object types. If both objects are people, an interaction is defined as either talking, walking together, or holding hands. If both objects are vehicles, an interaction is defined as one vehicle being either immediately behind or immediately in front of the other. If one object is a person and one a vehicle, then an interaction is defined as the person either driving/riding, entering, or exiting the vehicle.

their spatial configuration. For any set of vertices V in a graph, there is a corresponding set of rectangles $R = \{\text{box}(v), v \in V\}$, where $\text{box}(v)$ denotes the projection of the vertex v to its bounding box component in $\mathcal{R}$. To construct a distribution over sets of rectangles in $\mathbb{P}(\mathcal{R})$, we use the following model:

$$P(R; \lambda, u) = \frac{1}{Z} \exp\left[\sum_{k=1}^{K} \lambda_k u_k(R)\right],$$

where $R \subset \mathcal{R}$, and $\lambda = \{\lambda_1, \ldots, \lambda_K\}$ and $u = \{u_1, \ldots, u_K\}$ are parameters and features, respectively. The features used are described in section 6.7.

The conversion of an image into a set of bounding boxes is shown in figure 5.3. By converting our dataset of images, we have a dataset $\{R_1, \ldots, R_n\}$ of rectangle sets; this is used to learn the model. Some examples from this dataset are shown in figure 6.1 and some samples from the above model with learned parameters are shown in figure 6.2. We defer discussion of learning, sampling, and conditional sampling until after the random graph model is discussed.

6.2.1 Question Sequences

We use the distribution P for random rectangles to create sequences of unpredictable questions. Suppose we have an unknown rectangle set $R \in \mathbb{P}(\mathcal{R})$ that we want to uncover using questions. Let $\mathcal{F}$ be a question space with questions of the form $f : \mathbb{P}(\mathcal{R}) \to \{0, 1\}$; we will iteratively select questions from this space. On each iteration, we estimate the pre-

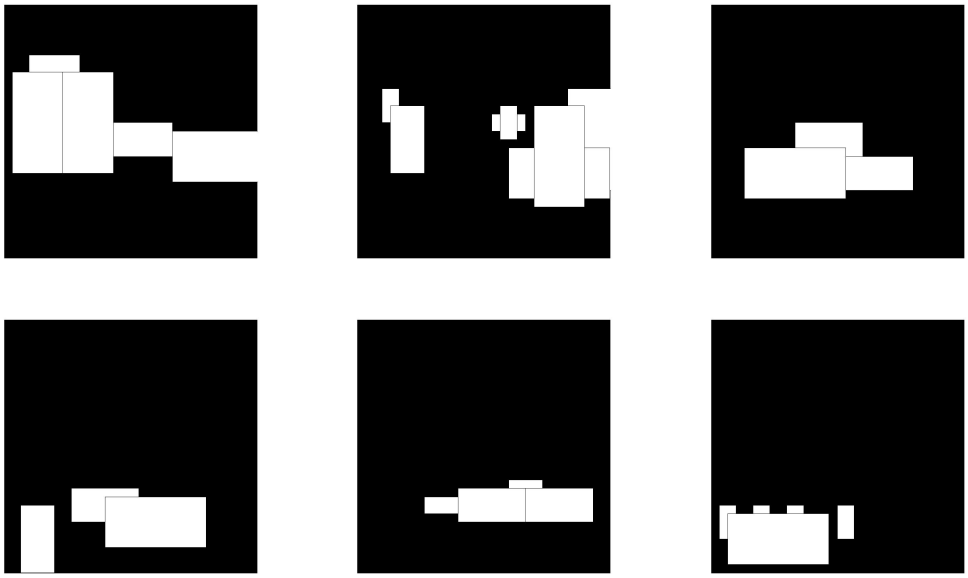


Figure 6.1: Examples from the dataset.

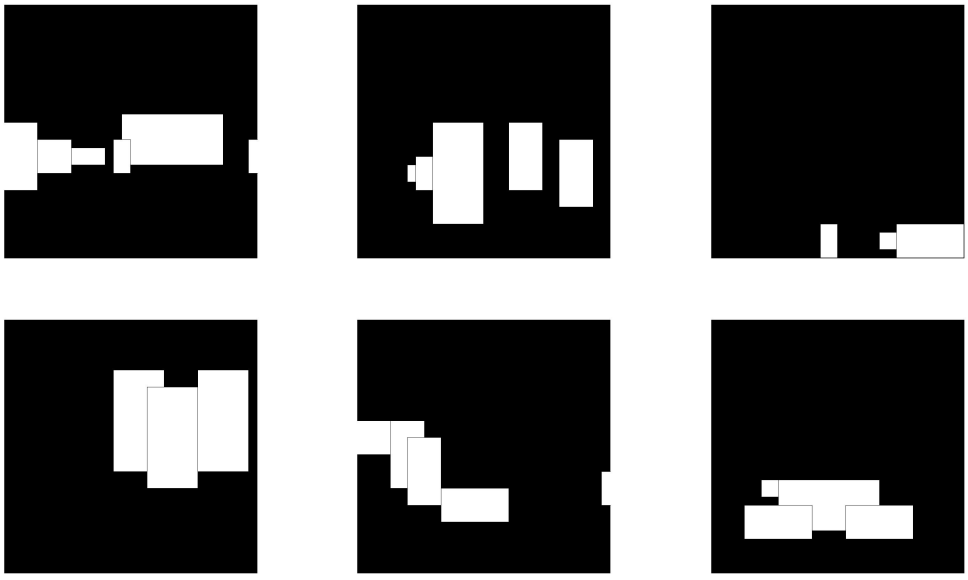


Figure 6.2: Samples from the model.

dictability of questions in $\mathcal{F}$, given the history of previously asked questions and answers, and then select a question that is ϵ-unpredictable. The predictabilities are estimated by sampling from the conditional distribution using the Metropolis-Hastings algorithm (details of this conditional sampling are provided in section 6.4). This sampling algorithm requires initial states (i.e., starting rectangle sets) that cohere with the current history; these are obtained by using the samples from the previous iteration that cohere with it. Details are shown in Algorithm 2 below.

An example sequence of unpredictable questions, generated using this algorithm, is displayed in appendix A.2.1. In this example, for the questions space $\mathcal{F}$, we use questions designed to instantiate rectangles. Looking ahead, we will associate people with vertical rectangles, and vehicles with horizontal ones, and the questions take the form:

1. "Is there an object of type t in region w?"

2. "Is there a unique object of type t in region w?"

These questions should be tacitly understood to include the qualifiers "partially visible" and "not previously instantiated." We display conditional samples that cohere with this sequence in figure 6.3; these samples can be seen to be constrained by the history.

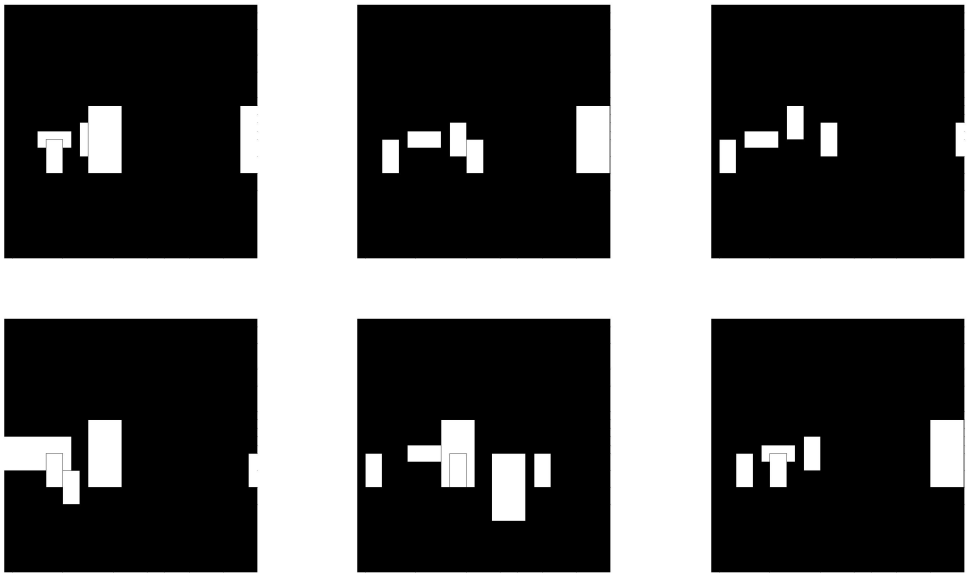


Figure 6.3: Conditional samples from the model, given the history $H_k = (F_k, A_k)$ shown in appendix A.2.1. Loosely, this history instantiates three people on the left-half of the image, instantiates a vehicle on the left-half, and establishes the absence of a vehicle in the bottom-right quadrant. People are associated with vertical rectangles, and vehicles associated with horizontal ones.

Algorithm 2 Forming Sequences of Unpredictable Questions

Input 1: A rectangle set $R_0 \in \mathbb{P}(\mathcal{R})$;

Input 2: A distribution P over $\mathbb{P}(\mathcal{R})$;

Output: A sequence $(f_1, \ldots, f_l)$ of unpredictable questions for R_0;

Let $H_0 = \emptyset$.

Generate a set $\mathcal{S}_0$ of rectangle sets by randomly sampling from $\mathbb{P}(\mathcal{R})$.

For $k = 1, \ldots, l$:

1. Generate m samples from the conditional distribution $P_{H_{k-1}}$; let $\mathbb{T}_k$ denote this set of samples. Each sample is generated as follows:

   (a) Randomly select a rectangle set $R \in \mathcal{S}_{k-1}$.

   (b) Perform Metropolis-Hastings algorithm (see section 6.4) to generate a sample, using the rectangle set R as an initial state.

2. Using the samples $\mathbb{T}_k$, estimate the set of unpredictable questions in $\mathcal{F}$:

   (a) For each question $f \in \mathcal{F}$, calculate its predictability:

   $$\hat{\rho}(f, H_{k-1}) = \left| \frac{\#\{R \in \mathbb{T}_k \mid f(R) = 1\}}{m} - 0.5 \right|.$$

   (b) Form the set of ϵ-unpredictable questions $\mathcal{U}_{H_{k-1}} = \{f \in \mathcal{F} \mid \hat{\rho}(f, H_{k-1}) \leq \epsilon\}$.

3. Randomly select a question $f_k \in \mathcal{U}_{H_{k-1}}$ and get its answer $a_k = f_k(R_0)$.

4. Set $H_k = H_{k-1} \cup (f_k, a_k)$.

   Set $\mathcal{S}_k = \{R \in \mathbb{T}_k \mid f_k(R) = a_k\}$.

6.3 Random Graphs

In the previous section, we considered the simplified problem of modeling the spatial configuration of objects in images. We now return to our original problem of modeling real-world scenes (i.e., developing a distribution over the graph space discussed in section 6.1). As before, once we have learned this distribution, we will use it estimate sequences of unpredictable questions.

To develop this model, we first constrict the graph space: let $U : \Lambda_V \times \Lambda_V \rightarrow \Lambda_E$ be a master interactions function that assigns the value 0 to every pair of vertices that cannot have an edge, and assigns a value 1 to every pair of vertices that can. Define U as follows:

$$U(v, v') = \begin{cases} 0, & \text{if } d(v, v') > t \\ 1, & \text{otherwise} \end{cases}$$

where $d(v, v')$ is the distance between the centers of the bounding boxes for the vertices. In other words, this master interactions function can be used to ensure there is no edge between vertices that are farther apart than t. Let the graph space be:

$$\mathcal{G} = \left\{ (V, E) \ : \ \begin{array}{l} V \subset \Lambda_V, \ |V| \leq N \\ E : V \times V \rightarrow \Lambda_E \\ E(v, v') \leq U(v, v') \text{ for all } v, v' \in V \end{array} \right\}.$$

where N is the maximum order allowed for graphs. To define a distribution over this space,

we use the chain rule and decompose the distribution into two parts. We do this so that, given a vertex in a graph, its attribute component (i.e., the attribute $A \subset \mathcal{A}_t$) does not depend on anything except that vertex's type $t \in \mathcal{T}$. Let's first define the projection of graphs in $\mathcal{G}$ to graphs without an attribute component in their vertices. For vertices $v = (t, A, r) \in \Lambda_V$, let $\pi(v) \equiv (t, r)$, and for graphs $G \in \mathcal{G}$, let $\pi(G) \equiv (\{\pi(v), v \in V\}, E)$. Finally, let $\mathcal{G}' = \{\pi(G) \mid G \in \mathcal{G}\}$ denote the projected space of graphs. Suppose there is a distribution P over the graph space $\mathcal{G}$; by the chain rule, we have that:

$$P(G) = P(G \mid \pi(\mathbf{G}) = G') \cdot P(\pi(\mathbf{G}) = G').$$

where $G \in \mathcal{G}$ and $G' = \pi(G)$. We model each component separately. Let μ_0 be a distribution over $\mathcal{G}'$ having the form:

$$\begin{aligned} \mu_0(G') &= \hat{P}(\pi(\mathbf{G}) = G') \\ &= \frac{1}{Z} \exp\left[\sum_{k=1}^{K} \lambda_k u_k(G')\right]. \end{aligned}$$

In contrast to the random rectangles model in the previous section, this model contains features that depend on the edges in the graph. The features are described in section 6.7.

Let $\mu_{G'}$ be a conditional distribution having the form:

$$\begin{aligned}\mu_{G'}(G) &= \hat{P}(G \mid \pi(\mathbf{G}) = G') \\ &= \frac{1}{Z} \prod_{v \in V(G)} \phi(\text{att}(v)),\end{aligned}$$

where ϕ is a real function over attributes $A \subset \mathcal{A}_t$ and att(v) is the projection of the vertex v to its attribute component. Thus, our model takes the form:

$$\hat{P}(G) = \mu_0(G')\mu_{G'}(G),$$

where $G \in \mathcal{G}$ and $G' = \pi(G)$.

Some examples from the dataset are shown in figure 6.4 and some samples from the model are shown in figure 6.5. The samples were generated using the Metropolis-Hastings algorithm given in section 3.1.

6.3.1 Question Sequences

We use the random graph model to create sequences of unpredictable questions. Suppose we have an unknown graph $G \in \mathcal{G}$ that we want to uncover and let $\mathcal{F}$ be a question space with questions of the form $f : \mathcal{G} \to \{0, 1\}$. As above, questions will be iteratively selected from this space; on each iteration, we estimate the predictability of questions in $\mathcal{F}$, given the history of previous questions and answers, and an ϵ-unpredictable question

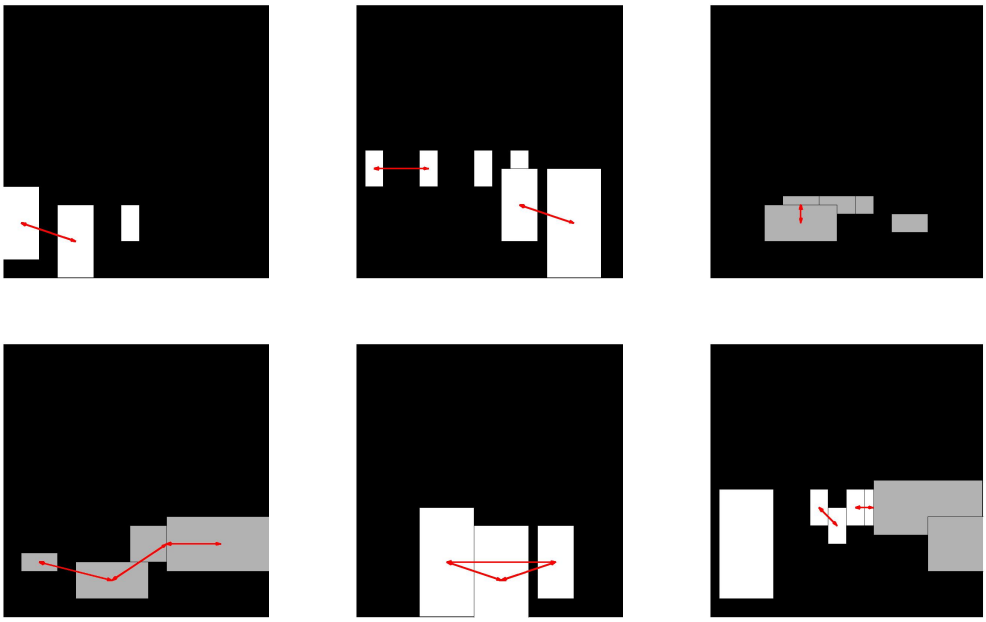


Figure 6.4: Examples from the dataset.

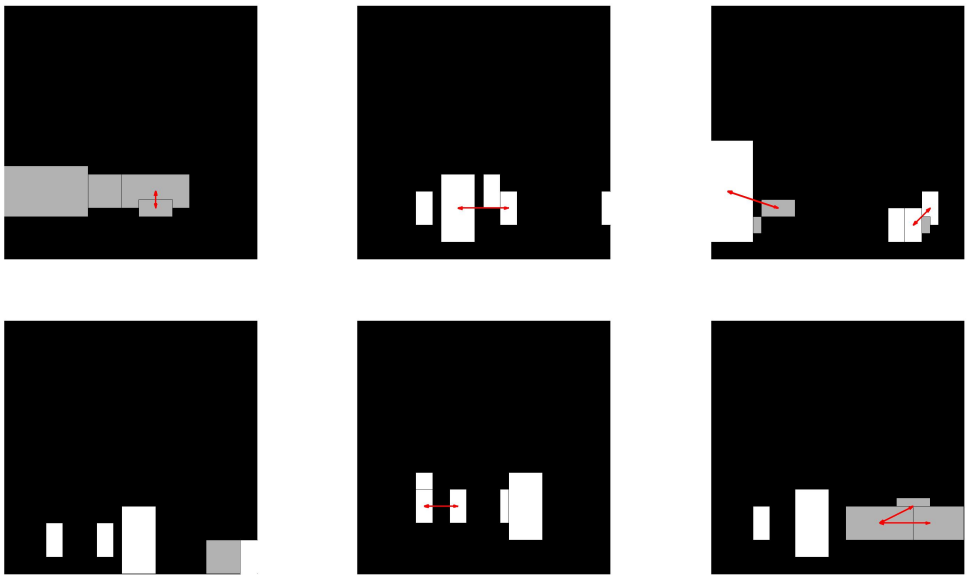


Figure 6.5: Samples from the model.

is selected. For additional detail, see Algorithm 2 above; the same algorithm is used here, except with the rectangle sets replaced with graphs.

An example sequence of unpredictable questions, generated using this algorithm for a particular graph G, is displayed in in appendix A.2.2. In this example, the question space $\mathcal{F}$ is composed of instantiation questions, attribute questions, and relationship questions:

1. "Is there an object of type t with attribute a in window w?"
2. "Is there a unique object of type t with attribute a in window w?"
3. "Does object k have attribute a?"
4. "Does object k have a relationship with any other object?"
5. "Is there a relationship between object k_1 and object k_2?"

Recall, the reference to an "object k" in these questions refers to the k^{th} object instantiated by a given history. The instantiation questions are tacitly assumed to include the qualifier "not previously instantiated". We display conditional samples given this sequence in figure 6.6.

6.3.2 Learning

The model μ_0 was learned using the stochastic learning algorithm given in section 3.2, except with some additional learning stages to initialize the parameters. We refer to features that are only functions of the vertices of a graph as *vertex features* (i.e. features $u_1, \ldots, u_{19}$

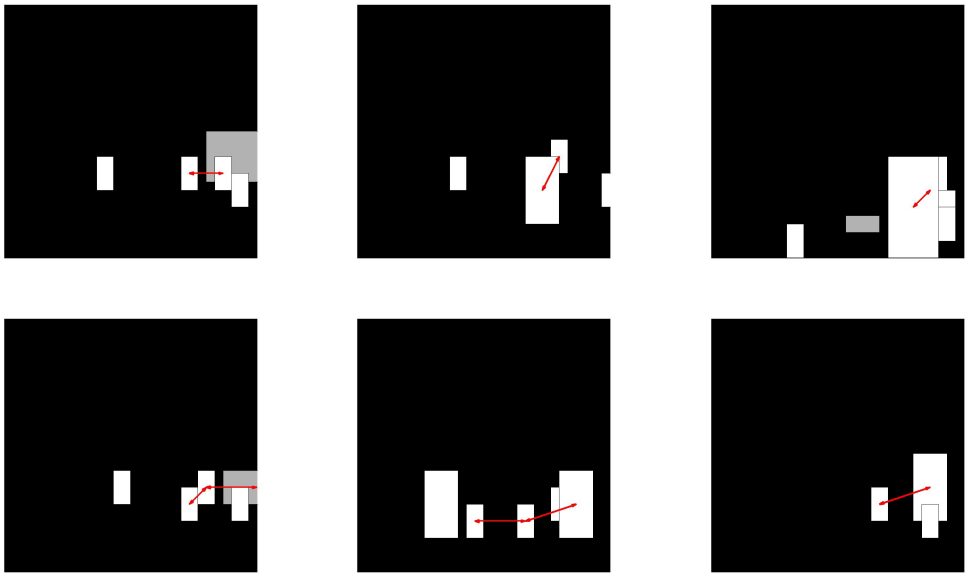


Figure 6.6: Conditional samples from the model, given the history $H_k = (F_k, A_k)$ shown in appendix A.2.2. Loosely, this history instantiates three people in the right-half of the image, two of which are interacting with each other, and establishes the absence of a vehicle in most of the image. People are associated with vertical white rectangles, and vehicles associated with horizontal grey ones.

in section 6.7), and refer to features that depend on the edges as *edge features* (i.e. features $u_{20}, \ldots, u_{22}$ in section 6.7). We perform the learning in three steps.

1. (Initialize the vertex feature parameters) Perform stochastic learning for the parameters of the vertex features, keeping other parameters fixed.

2. (Initialize the edge feature parameters) Perform stochastic learning for the parameters of the edge features, keeping other parameters fixed.

3. Perform stochastic learning on all parameters.

We used $M = 250$ particles and $T = 20,000$ iterations for each of the three steps above. The learning rate α_t was kept fixed at 0.01 for the first 5000 parameter updates, and was

then reduced as $10/(100+t)$.

6.4 Graph Conditional Sampling

Let $\mathcal{G}$ denote a finite graph space and let P be a distribution over it. In this section, we consider conditional sampling from this distribution, where the evidence will take the form of a sequence of questions and answers. Suppose we have a history $H = (F, A)$, and let $\mathcal{G}_H$ be the set of graphs that coheres with this history:

$$\mathcal{G}_H \equiv \{G \in \mathcal{G} \mid F(G) = A\}.$$

We want to sample from conditional distributions of the form $P_H(G) \equiv P(G \mid \mathcal{G}_H)$. We will use the Metropolis-Hastings algorithm, which we repeat here for convenience:

Algorithm: (Metropolis-Hastings) Given a *transition kernel* $q_H(G'|G)$ and starting from an initial state G_1, repeat the following steps from $t = 1$ to T:

1. Generate a candidate $G' \sim q_H(G'|G_t)$.

2. Generate $U \sim \mathcal{U}(0,1)$ and set

$$G_{t+1} = \begin{cases} G', & \text{if } U \leq \alpha(G_t, G') \\ G_t, & \text{otherwise} \end{cases}$$

where $\alpha(G, G')$ is the *acceptance probability*, given by:

$$\alpha(G, G') = \min \left\{ \frac{P_H(G')\ q_H(G\ |G')}{P_H(G)\ q_H(G'\ |G)}, 1 \right\}.$$

This algorithm will generate a sequence $G_1, G_2, \ldots$ of graphs. To ensure that this chain has the stationary distribution P_H, it suffices for it to be ergodic. Since the graph space is finite, a chain is ergodic if there exists a number n such that the chain can go from any graph to any other graph in exactly n steps with positive probability. We can guarantee ergodicity by checking two conditions: (1) the chain can reach any graph in the set $\mathcal{G}_H$ from any other graph in this set with positive probability; and (2) for any graph in this set, there is a positive probability of returning to itself in one step. Thus, the transition kernel must be designed to meet these requirements.

Before considering the kernel, however, notice that the questions in the history H affect the difficulty of conditional sampling. If questions can be arbitrary (i.e., questions can be any function $f : \mathcal{G} \to \{0, 1\}$), then the space $\mathcal{G}_H$ can be convoluted and disconnected with respect to the simple, first-order moves we discuss below. Indeed, with arbitrary questions, the space $\mathcal{G}_H$ can be an arbitrary set, and hence conditional sampling would be almost impossible without a brute force approach. Thus, we restrict our attention to the questions used in the previous chapter. In particular, we only consider instantiation questions in $\mathcal{F}_{\text{inst}}$ rather than in $\tilde{\mathcal{F}}_{\text{inst}}$; these questions have the valuable property that vertices can only be instantiated once (see chapter 4). Notice that if a history has questions from $\tilde{\mathcal{F}}_{\text{inst}}$, then the

space $\mathcal{G}_H$ can be disconnected with respect to first-order moves. For example, suppose a sequence has two uniqueness questions $f_{B_1}, f_{B_2} \in \tilde{\mathcal{G}}_{\text{uniq}}$, where $B_1 \neq B_2$ and $B_1 \cap B_2 \neq \emptyset$, and both questions have positive answers. Then, the unknown graph either has one vertex in $B_1 \cap B_2$, or two vertices, one in $B_1 \setminus (B_1 \cap B_2)$ and the other in $B_2 \setminus (B_1 \cap B_2)$. This is a problem because there is no way to transition between these two scenarios with first-order moves (i.e., loosely, moves where a proposed graph can only differ from the current graph by one vertex).

6.4.1 Transition Kernel

Let's now consider the transition kernel. Given the current graph $G \in \mathcal{G}_H$ in a chain, a proposal graph $G' \in \mathcal{G}_H$ is sampled according to the kernel. In chapter 3, we discussed a generic kernel based on the following moves (we define them more formally below):

1. Adding a vertex to G.
2. Deleting a vertex from G.
3. Changing an edge value in G.

However, this set of moves is insufficient for irreducibility of the chain in $\mathcal{G}_H$. To see this, notice that the first two moves (deleting or adding a vertex) can never allow adjustments to a vertex that has been instantiated[2] (while staying within the set $\mathcal{G}_H$). For this reason, we

[2]Suppose we have instantiated a vertex using the set $B \subset \Lambda_V$. The instantiated vertex can not be deleted and a vertex can never be added that is in the set $B \subset \Lambda_V$ (either move would violate the history), hence the instantiated vertex can never be adjusted (i.e. take a different value in B).

add a fourth move:

4 Changing a vertex in G. (That is, a vertex $v \in V(G)$ is changed by deleting vertex v and edges incident on it from the graph, then adding another vertex v' and edges incident on it to the graph.)

With this set of moves, any graph in $\mathcal{G}_H$ can be reached from any other one. We define the above moves more formally. Given a graph $G \in \mathcal{G}_H$, the set of possible proposal graphs for each type of move are given by the following sets:

$$\mathcal{G}_H^{\text{add}}(G) \equiv \{G' \in \mathcal{G}_H \mid V' = V \cup \{v\}, v \in \Lambda_V \text{ and } E = E' \restriction_{V \times V}\}$$

$$\mathcal{G}_H^{\text{del}}(G) \equiv \{G' \in \mathcal{G}_H \mid V' = V \setminus \{v\}, v \in V \text{ and } E' = E \restriction_{V \setminus \{v\} \times V \setminus \{v\}}\}$$

$$\mathcal{G}_H^{\text{cha}}(G) \equiv \{G' \in \mathcal{G}_H \mid G' \in \mathcal{G}_H^{\text{add}}(\mathcal{G}_H^{\text{del}}(G))\}$$

$$\mathcal{G}_H^{\text{edge}}(G) \equiv \{G' \in \mathcal{G}_H \mid V' = V \text{ and } E' = E \text{ except for one pair of vertices.}\}$$

A move is selected with some probability, and then a proposal graph is randomly selected from that move's set.

6.4.2 Simplifications

We notice that the above set of moves is slightly more general than needed and we find it convenient to simplify them. Suppose, for a given graph G, a vertex v_0 has been instantiated. This implies that we have identified a set $B \subset \Lambda_V$ such that $v_0 \in B$ and $v \notin B$

for all $v \in V \setminus \{v_0\}$. Further, suppose that we have asked some relationship questions about vertex v_0 and have established edge values between v_0 and vertices $v_1, \ldots, v_l$. In this scenario, every graph that coheres with the history (i.e. every graph in $\mathcal{G}_H$) must have one and only one vertex in $B \subset \Lambda_V$ with a similar arrangement of edges present. That is, if the chain performs a simple move of changing vertex v_0 to another vertex $v_0' \in B$, we require that vertex v_0' has the same edge arrangement as v_0. Another way of saying this is that we require the proposal graph G' to be isomorphic to G for these types of moves. More formally, we need these graphs to be directly isomorphic:

Definition (Direct Isomorphism): A graph $G = (V, E)$ is *directly isomorphic* to the graph $G' = (V', E')$ if there exists a bijection $f : V \to V'$ such that:

1. $f(v) = v$ for all $v \in V \cap V'$.

2. $E(v, v') = E'(f(v), f(v'))$ for all $v, v' \in V$.

Two graphs that are directly isomorphic are denoted by $G \simeq_d G'$.

In other words, this definition requires that any vertex that is in both graphs gets mapped to itself. With this definition in hand, we simplify the proposal set for this type of move to the following:

$$\tilde{\mathcal{G}}_H^{\text{cha}}(G) \equiv \{G' \in \mathcal{G}_H \mid G' \in \mathcal{G}_H^{\text{add}}(\mathcal{G}_H^{\text{del}}(G)) \text{ and } G \simeq_d G'\}.$$

We also simplify the deletion of vertices to only vertices with no edges, and simplify the addition of vertices to have no incident edges on the new vertex. That is, we simplify the proposal sets for these types of moves to the following:

$$\tilde{\mathcal{G}}_H^{\text{add}}(G) \equiv \{G' \in \mathcal{G}_H \mid V' = V \cup \{v\}, v \in \Lambda_V \text{ and } E = E' \restriction_{V \times V} \text{ and } E' \restriction_{\{v\} \times V} = 0\}$$

$$\mathcal{G}_H^{\text{del}}(G) \equiv \{G' \in \mathcal{G}_H \mid V' = V \setminus \{v\}, E(v, \cdot) = 0, v \in V \text{ and } E' = E \restriction_{V \setminus \{v\} \times V \setminus \{v\}}\}.$$

These simplifications help the efficiency of the chain while maintaining irreducibility.

6.4.3 Computation

Suppose we have a history H and want to sample from P_H. We consider the computations necessary for performing each type of move.

1. Addition of Vertices: For computational efficiency, we pre-compute the set of vertices that can be added to a graph. Let $\Lambda_V^{\text{add}}(H, G) \subset \Lambda_V$ denote the set of vertices that can be added to the graph G while not violating the history H. That is, let

$$\Lambda_V^{\text{add}}(H, G) \equiv \{v \in \Lambda_V \mid G_{+v} \in \mathcal{G}_H\}$$

where G_{+v} denotes the graph in $\tilde{\mathcal{G}}^{\text{add}}(G)$ that corresponds to the addition of the vertex v to the graph G. Notice, due to the properties of the instantiation questions in $\mathcal{F}_{\text{inst}}$, we have

that

$$\Lambda_V^{\text{add}}(H, G) = \Lambda_V^{\text{add}}(H) \text{ for all } G \in \mathcal{G}_H.$$

Hence, the set of vertices $\Lambda_V^{\text{add}}(H)$ that may be added to a graph does not depend on the particular graph G and can be pre-computed[3] (with the exception that vertices already in the graph cannot be added to it). This is important since this set will be used numerous times in any chain.

2. Deletion of Vertices: The set of vertices that can be deleted from a given graph G while staying coherent with a history H is graph dependent; this set has to be kept up-to-date for the current graph in any chain. Note that this set cannot be determined by looking at each vertex independently; it must be determined from all vertices.

3. Change of Vertices: In a given graph G, the set of possible vertices that a vertex v can be changed to, while still cohering with the history, is not the same as the set of possible vertex additions $\Lambda_V^{\text{add}}(H)$ given above. (Instantiated vertices must be allowed to be adjusted.) Let $\Lambda_V^{\text{cha}}(v, H, G) \subset \Lambda_V$ denote the set of vertices that the vertex $v \in V(G)$ can be changed to

[3]The set of vertex additions $\Lambda_V^{\text{add}}(H)$ does not depend on the graph G due to the nature of the questions in our question space. Suppose we have a graph $G \in \mathcal{G}_H$ that satisfies the history. Then, the addition of a vertex to G can cause a violation of the history if: (a) there is an existence question with a negative answer, or (b) there is a uniqueness question with a positive answer, and the object overlaps the region in one of these questions.

in the graph G while not violating the history H. That is, let

$$\Lambda_V^{\text{cha}}(v, H, G) \equiv \{v' \in \Lambda_V \mid G_{v \to v'} \in \mathcal{G}_H\}$$

where $G_{v \to v'}$ denotes the graph in $\tilde{\mathcal{G}}^{\text{cha}}(G)$ that corresponds to the deletion of vertex v and the addition of the vertex v' to the graph G.

We notice that this set can be computed efficiently by pre-computing a superset that is graph independent and then pruning based on the particular graph G in the chain and the vertex in it to adjust. That is, we can pre-compute[4] a set $\Lambda_V^{\text{cha}}(H) \subset \Lambda_V$ that only depends on the history such that:

$$\Lambda_V^{\text{cha}}(v, H, G) \subseteq \Lambda_V^{\text{cha}}(H),$$

for any graph G and vertex v in it, then prune it as necessary. As with vertex deletion, the set $\Lambda_V^{\text{cha}}(v, H, G)$ must be determined using the entire graph due to complicated interdependencies between the vertices and the sequence of questions and answers.

4. Change of Edges: In our model, a pair of vertices v, v' can have an edge only if $d(v, v') < t$. Hence, the set of pairs of vertices that can have an edge has to be kept up-to-date for the current graph in any chain. This set is also used in the calculations for

[4]The set $\Lambda_V^{\text{cha}}(H)$ is created using the following observation. In general, the only change of a vertex v to another vertex v' that always causes a violation of the history is if v' is in a region in which an existence question has a negative answer.

the change of vertices move (since the new vertex must stay within t of any vertex it has an edge with).

6.5 Analysis of Graph Conditional Sampling

To check the validity of our conditional sampling algorithm, we test whether or not these conditional samples are correctly distributed. Suppose we have a given history H_k of length $k = 10$; then we form two datasets of graphs as follows.

1. The first dataset $\mathcal{G}_0$ is formed by sampling according to the distribution P_{H_k} using the conditional sampling algorithm in the previous section.

2. The second dataset $\mathcal{G}_1$ is formed by: (a) sampling a million samples according to the distribution P; and (b) filtering these samples to only those that cohere with the history H_k.

By design, the dataset $\mathcal{G}_1$ is distributed according to the true conditional distribution. To test the validity of the conditional sampling algorithm, we compare $\mathcal{G}_0$ to $\mathcal{G}_1$ by:

1. comparing the histograms for the likelihoods of the samples (see figure 6.7).

2. comparing the mean value of scene features in the datasets (see tables 6.1, 6.2, and 6.3).

Although the conditional sampling algorithm should work in theory, this analysis indicates it also works surprisingly well in practice (or, to be more precise, doesn't show any obvious

signs of being amiss).

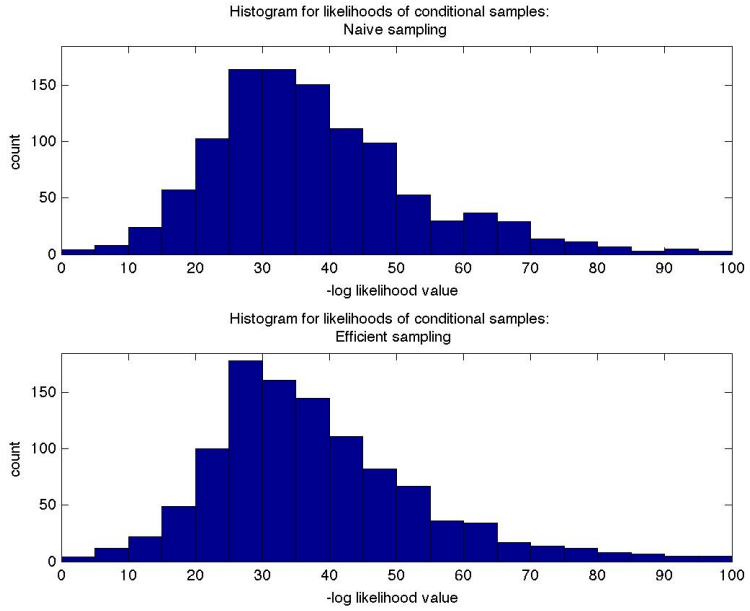


Figure 6.7: The top figure shows the (unnormalized) likelihood histogram for the set $\mathcal{G}_1$ and the bottom figure for the set $\mathcal{G}_0$. The size of the datasets are $|\mathcal{G}_1| = |\mathcal{G}_0| = 1090$, and a randomly generated history was used for forming these sets.

6.6 Analysis of Model Validity

In the previous section, we tested the validity of the conditional sampling algorithm. In this section, we test the validity of our scene model with respect to the task of selecting unpredictable questions. Suppose we have a graph G and we generate a sequence of questions $(f_1, \ldots, f_l)$ for it using our statistical model. For each selected question f_k, we will record the estimated conditional probability of that question having a positive answer, as well as its actual answer $a_k = f_k(G)$. If we collect this data over enough (randomly selected) graphs, then by comparing the estimated number of positive answers to the observed number of positive answers, we can assess the accuracy of the model's conditional probability estimates, and in turn, the accuracy of the model in selecting ϵ-unpredictable questions.

The results are shown in the tables 6.4, 6.5, and 6.6. These tables indicate that the predictability of questions according to the model often differs from their true predictability, yet on average, the model selects questions that fall within the acceptable range for being considered unpredictable, i.e. on average, the questions are unpredictable within a tolerance of $\epsilon = 0.15$.

6.7 Features

6.7.1 Random Rectangle Features

For the random rectangle model in section 6.2, we used the following 14 features:

1. (Number of people and vehicles in a scene).

$$u_1(R) = |N_{\text{horz}}(R) + N_{\text{vert}}(R) - 6|.$$

where

$$N_{\text{horz}}(R) = \text{\# of horizontal rectangles in } R.$$

$$N_{\text{vert}}(R) = \text{\# of vertical rectangles in } R.$$

The expected number of objects in an image (according to the training dataset) is 6.

This feature controls the variation around the expected number of objects.

2. (Amount of overlap)

$$u_2(R) = \sum_{\substack{r_i, r_j \in R \\ r_i \neq r_j}} I_{\text{overlap}(r_i, r_j) > 0.75}$$

where

$$\text{overlap}(r_i, r_j) = \frac{\text{area}(r_i \cap r_j)}{\min(\text{area}(r_i), \text{area}(r_j))}.$$

Although it is common for the annotated objects in our dataset to overlap, it is rare that they overlap completely; this feature penalizes objects that overlap too much. The value of 0.75 used in the feature was chosen by hand and is somewhat arbitrary.

3. (Location/size combinations of objects)

$$u_{z,s}(R) = \sum_{r \in R} I_{\text{center}(r) \in z} \; I_{\text{size}(r)=s}$$

where z is an image region (either the top-third, middle-third, or bottom-third of an image), and s is an object size (either small, medium, or large). Hence, we are using 9 features based on the 9 possible location and size combinations. Due to the camera perspective, larger objects (in terms of image pixels) tend to occur more often in the bottom of the image than at the top of the image. This feature allows these types of relationships to be encoded in the model.

4. (Larger object above smaller object?)

$$u_{12}(R) = \frac{1}{|R|(|R|-1)/2} \sum_{\substack{r_i, r_j \in R \\ r_i \neq r_j}} I_{\{r_i \text{ is above } r_j \text{ and has a larger size.}\}}$$

Similar to the above feature, the typical camera perspective tends to make larger objects appear at the bottom of the image and smaller ones at the top. This feature can penalize scenes that violate this.

5. (Objects tend to "line up" horizontally)

$$u_{13}(R) = \frac{1}{|R|(|R|-1)} \sum_{r \in R} \sum_{\substack{r' \in R \\ r' \neq r}} I_{\{\text{A horizontal line through the center of } r \text{ passes through } r'\}}$$

$$u_{14}(R) = \frac{1}{|R|(|R|-1)/2} \sum_{\substack{r_i, r_j \in R \\ r_i \neq r_j}} I_{\{\text{A horizontal line passes through both } r \text{ and } r'\}}$$

Often in our imagery, objects occur on a similar horizontal level. This phenomena can be observed in the dataset; for example, it can be observed in the examples shown in figure 6.4.

6.7.2 Random Graph Features

For the random graph model from section 6.3, we used the 14 features from the previous section, plus the following 8 features:

1. (Number of people and vehicles in a scene).

$$u_{15}(G) = |N_{\text{people}}(G) - 3|$$

$$u_{16}(G) = |N_{\text{vehicle}}(G) - 3|,$$

where

$$N_{\text{people}}(G) = \text{\# of people in } G.$$

$$N_{\text{vehicle}}(G) = \text{\# of vehicles in } G.$$

These features are similar to feature u_1 above. The expected number of people in an image is roughly 3 and the expected number of vehicles in an image is roughly 3, and these features control the variation around these expected values.

2. (Distances between objects)

$$u_{17}(G) = |N_t^1(G) - \rho_1 N_\infty^1(G)|$$

$$u_{18}(G) = |N_t^2(G) - \rho_2 N_\infty^2(G)|$$

$$u_{19}(G) = |N_t^3(G) - \rho_3 N_\infty^3(G)|,$$

where

$$N_t^1(G) = \sum_{\substack{v_i,v_j \in V \\ v_i \neq v_j}} I_{\{\text{type}(v_i)=\text{type}(v_j)=\text{`person'}\}} I_{\{d(v_i,v_j)<t\}}$$

$$N_t^2(G) = \sum_{\substack{v_i,v_j \in V \\ v_i \neq v_j}} I_{\{\text{type}(v_i)=\text{type}(v_j)=\text{`vehicle'}\}} I_{\{d(v_i,v_j)<t\}}$$

$$N_t^3(G) = \sum_{\substack{v_i,v_j \in V \\ v_i \neq v_j}} I_{\{\text{type}(v_i)\neq\text{type}(v_j)\}} I_{\{d(v_i,v_j)<t\}}$$

and $d(v, v')$ is the distance between the centers of the two objects. (We set $t = 4$, $\rho_1 = 0.56$, $\rho_2 = 0.44$, and $\rho_3 = 0.36$.) These features control the distances between the objects; to be more specific, they control the amount of variation of the number of objects close to each other in an image (i.e., closer than distance t) relative to their expected number.

3. (Number of edges)

$$u_{20}(G) = \sum_{\substack{v_i,v_j \in V \\ v_i \neq v_j}} I_{\{\text{type}(v_i)=\text{type}(v_j)=\text{`person'}\}} I_{\{d(v_i,v_j)<t\}} I_{\{E(v_i,v_j)=1\}}$$

$$u_{21}(G) = \sum_{\substack{v_i,v_j \in V \\ v_i \neq v_j}} I_{\{\text{type}(v_i)=\text{type}(v_j)=\text{`vehicle'}\}} I_{\{d(v_i,v_j)<t\}} I_{\{E(v_i,v_j)=1\}}$$

$$u_{22}(G) = \sum_{\substack{v_i,v_j \in V \\ v_i \neq v_j}} I_{\{\text{type}(v_i)\neq\text{type}(v_j)\}} I_{\{d(v_i,v_j)<t\}} I_{\{E(v_i,v_j)=1\}}.$$

These features control the tendency of various objects to have edges with each other.

	$\mathcal{G}_1$	$\mathcal{G}_0$
# people	5.7174	5.7220
# vehicles	1.1009	1.1018
feature1	1.2495	1.3303
feature2	0.5742	0.6289
feature3	0.1248	0.1073
feature4	0.0615	0.0404
feature5	0.0615	0.0697
feature6	4.7651	4.8459
feature7	1.1624	1.1853
feature8	0.4367	0.4183
feature9	0.1477	0.1798
feature10	0.0358	0.0330
feature11	0.0229	0.0284
feature12	0.0923	0.1149
feature13	0.7839	0.7802
feature14	0.9419	0.9383
feature15	2.9474	2.9566
feature16	2.0767	2.1193
feature17	1.5795	1.5870
feature18	0.2324	0.3034
feature19	0.9336	1.0302
feature20	2.3211	2.3284
feature21	0.1881	0.2954
feature22	0.4532	0.4963

Table 6.1: The mean value for scene features in each dataset. The features correspond to the features used in the model and are shown in section 6.7. The size of the datasets are $|\mathcal{G}_1| = |\mathcal{G}_0| = 1090$, and a randomly generated history was used for forming these sets.

	$\mathcal{G}_1$	$\mathcal{G}_0$
# male	2.4083	2.3908
# female	3.3092	3.3358
# children	0.1991	0.2147
# adults	5.5183	5.5119
# walking on sidewalk	1.8798	1.8661
# crossing street	1.4275	1.4358
# standing still	1.5147	1.5394
# entering or exiting a building	0.1055	0.0972
# sitting	0.7899	0.7881
# wearing hat	0.9853	0.9156
# not wearing hat	4.7321	4.8110
# carrying something	1.7468	1.5881
# not carrying something	3.9706	4.1385

Table 6.2: The mean value for scene features involving the attributes of people in each dataset. The size of the datasets are $|\mathcal{G}_1| = |\mathcal{G}_0| = 1090$, and a randomly generated history was used for forming these sets.

	$\mathcal{G}_1$	$\mathcal{G}_0$
# moving	0.4670	0.5046
# stopped	0.1596	0.1798
# parked	0.4743	0.4972
# car	0.8761	0.9376
# truck	0.0450	0.0541
# bus	0.0422	0.0642
# motorcycle	0.0633	0.0468
# bicycle	0.0743	0.0789
# no tire	0.2376	0.2615
# one tire	0.2596	0.2917
# two tires	0.4486	0.5000
# three or more	0.1550	0.1284
# dark color	0.4881	0.4945
# light color	0.4642	0.5174
# in-between color	0.1477	0.1697

Table 6.3: The mean value for scene features involving the attributes of vehicles in each dataset. The size of the datasets are $|\mathcal{G}_1| = |\mathcal{G}_0| = 1090$, and a randomly generated history was used for forming these sets.

estimated #yes	observed #yes
553.4 (47.1%)	475.0 (40.4%)

Table 6.4: We randomly sampled $n = 47$ images; on each image, we ran the query-engine to generate a sequence of length $k = 25$. Hence, in total, there are 1175 questions and answers.

	estimated #yes	observed #yes
questions 1-5	112.8 (48.0%)	103 (43.8%)
questions 6-10	108.5 (46.2%)	87 (37.0%)
questions 11-15	110.7 (47.1%)	92 (39.2%)
questions 16-20	109.1 (46.4%)	110 (46.8%)
questions 21-25	112.3 (47.8%)	83 (35.3%)

Table 6.5: This table shows the accuracy of the model in producing unpredictable questions based on the question's position in the sequence. We randomly sampled $n = 47$ images; on each image, we ran the query-engine to generate a sequence of length $k = 25$. Hence, in total, there are 1175 questions and answers; each row in row in the table has 235 questions and answers.

	estimated #yes	observed #yes
instantiation questions	335.1 (48.4%)	272 (39.3%)
attribute questions	161.2 (45.0%)	152 (42.5%)
relationship questions	57.1 (46.1%)	51 (41.1%)

Table 6.6: This table shows the accuracy of the model in producing unpredictable questions based on the question's type: instantiation, attribute, or relationship question. We randomly sampled $n = 47$ images; on each image, we ran the query-engine to generate a sequence of length $k = 25$. Hence, in total, there are 1175 questions and answers.

Appendix A

Question Streams

A.1 Non-rigorous Estimation

In this section, we display three examples of question sequences produced from our implementation of the visual Turing test in chapter 5. For readability, we display the questions in a colloquial form, but localization questions are understood to include the qualifier "partially visible" and instantiation questions are understood to include the qualifier "not previously instantiated." To the right of the localization questions we have inserted a small copy of the image, delineating the designated region with thick blue lines. (Occasionally, the blue lines make the determination of "partially visible" appear ambiguous. This is an artifact of their thickness, used here for better visibility.) Objects that would be annotated in the training set are highlighted with thin-lined, yellow bounding boxes.

As noted in chapter 5, the human operator serves two purposes, rejecting ambiguous

questions and providing correct answers for unambiguous questions. For an example of the occasional ambiguous question, see question 24 (Q24) in the first sequence, which asks if the lady on the right-hand side of the image is taller than the the man standing on the left-hand side. Another example is question 34 (Q34) which asks if that same lady is taller than the man sitting on the left-hand side. Most people would answer no to each comparison, but without certainty. In the second sequence, see questions Q18 and Q64, which also inquire about the sizes of objects, and were also declared ambiguous. For an additional example of an ambiguous question, see question 4 (Q4) of the third sequence, which asks "Is person1 carrying something?" It is impossible to tell since the right arm of person 1 is hidden by the male in the foreground. The query engine then excludes the question and continues the algorithm from the point where it left off, which in this case was inside the attribute loop that immediately followed the instantiation of person 1.

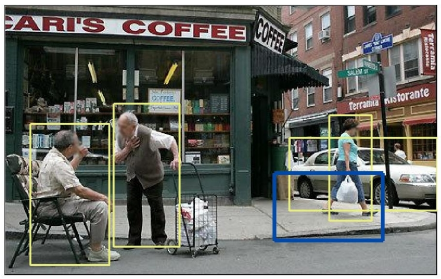
One of the design considerations was the level of difficulty—we wanted to avoid query streams that would completely defeat state-of-the-art vision systems. Still, there is plenty of challenge, as illustrated by the second sequence. The child sitting on the bicycle severely occludes the male adult; a small number of pixels representing the basket of the front bicycle determine that the child is partially occluded; and some of the questions require distinguishing bicycles that are moving from bicycles that are stopped.

A.1.1 Example 1

---

1. **Q:** Is there a person in the designated region?
   A: yes

2. **Q:** Is there a unique person in the designated region?
   A: yes (person1)



---

3. **Q:** Is person1 carrying something?
   A: yes

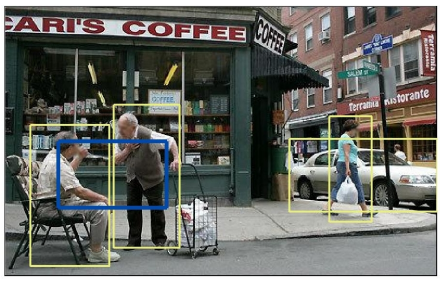
4. **Q:** Is person1 female?
   A: yes

5. **Q:** Is person1 walking on a sidewalk?
   A: yes

6. **Q:** Is person1 interacting with any other object?
   A: no

---

7. **Q:** Is there a vehicle in the designated region?
   A: no



8. **Q:** Is there a vehicle in the designated region?
   A: yes

9. **Q:** Is there a unique vehicle in the designated region?
   A: yes (vehicle1)

---

10. **Q:** Is vehicle1 light-colored?
    A: yes

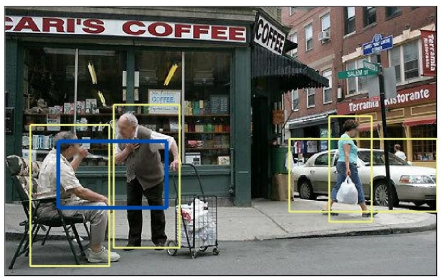
11. **Q:** Is vehicle1 moving?
    A: no

12. **Q:** Is vehicle1 parked and a car?
    A: yes

13. **Q:** Does vehicle1 have no visible tires or have three or more visible tires?
    A: no

14. **Q:** Does vehicle1 have one visible tire?
    A: no

15. **Q:** Is vehicle1 interacting with any other object?
    A: no



16. **Q:** Is there a person in the designated region?
    A: yes

17. **Q:** Is there a unique person in the designated region?
    A: no

18. **Q:** Is there a unique person that is female in the designated region?
    A: no

19. **Q:** Is there a person that is standing still in the designated region?
    A: yes

20. **Q:** Is there a unique person that is standing still in the designated region?
    A: yes (person2)

---

21. **Q:** Is person2 male?
    A: yes

22. **Q:** Is person2 carrying something?
    A: no

23. **Q:** Is person2 interacting with any other object?
    A: yes

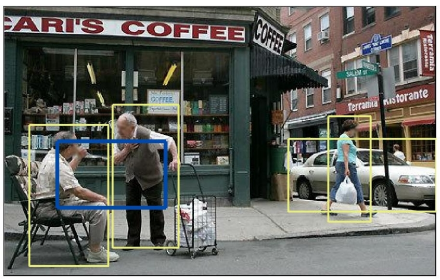
24. **Q:** Is person1 taller than person2?
    A: ambiguous

25. **Q:** Is person1 closer (to the camera) than person2?
    A: no

---

26. **Q:** Is there a person in the designated region?
    A: yes

27. **Q:** Is there a unique person in the designated region?
    A: yes (person3)



---

28. **Q:** Is person3 carrying something?
    A: no

29. **Q:** Is person3 walking on a sidewalk?
    A: no

30. **Q:** Is person3 female?
    A: no

31. **Q:** Is person3 standing still?
    A: no

32. **Q:** Is person3 an adult and crossing a street?
    A: no

33. **Q:** Is person3 interacting with any other object?
    A: yes

34. **Q:** Is person1 taller than person3?
    A: ambiguous

35. **Q:** Is person1 closer (to the camera) than person3?

A: no

36. **Q:** Is there an interaction between person2 and person3?
A: yes

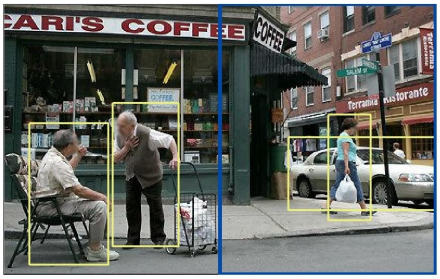
37. **Q:** Are person2 and person3 talking?
A: yes

38. **Q:** Is person2 taller than person3?
A: no

39. **Q:** Is person2 closer (to the camera) than person3?
A: no

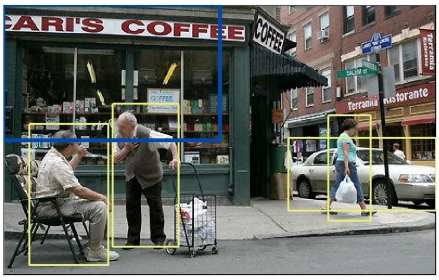
---

40. **Q:** Is there a vehicle in the designated region?
A: no



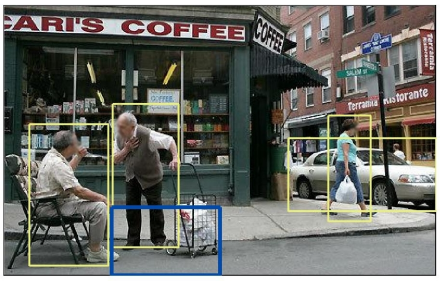
---

41. **Q:** Is there a vehicle in the designated region?
A: no

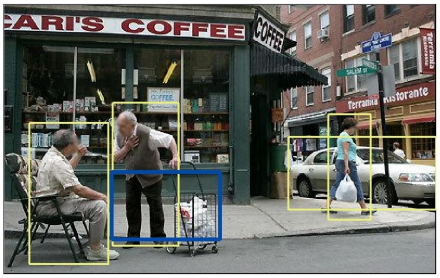


---

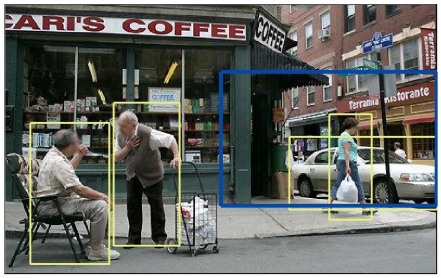
42. **Q:** Is there a vehicle in the designated region?
A: no



43. **Q:** Is there a person in the designated region?
A: no



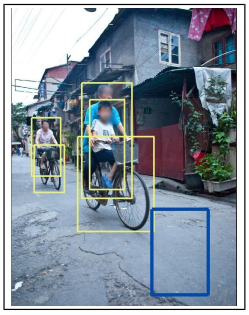
44. **Q:** Is there a person in the designated region?
A: no



A.1.2 Example 2

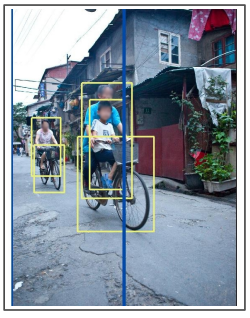
---

1. Q: Is there a person that is not wearing a hat in the designated region?
A: no



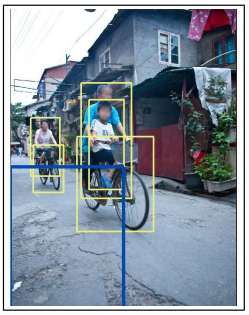
---

2. Q: Is there a person that is female in the designated region?
A: yes



---

3. Q: Is there a unique person that is female in the designated region?
A: yes (person1)



---

4. Q: Is person1 not carrying something?
A: yes

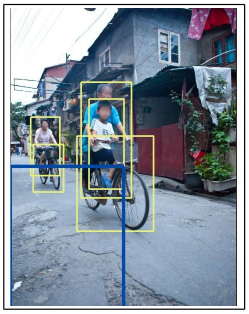
5. Q: Is person1 crossing a street or walking on a sidewalk?
A: no

6. Q: Is person1 not wearing hat and standing still?
A: no

7. Q: Is person1 interacting with any other object?
A: yes

---

8. Q: Is there a person that is not wearing a hat in the designated region?
A: yes

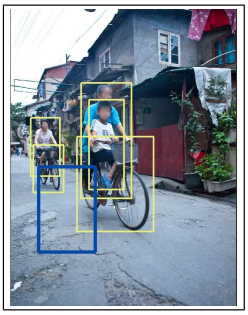


9. Q: Is there a unique person that is male, not wearing a hat in the designated region?
A: no

10. Q: Is there a person that is female, adult in the designated region?
A: no

---

11. Q: Is there a unique person that is adult, not wearing a hat in the designated region?
A: yes (person2)



---

12. Q: Is person2 walking on a sidewalk or sitting?
A: yes

13. Q: Is person2 not carrying something?

A: yes

14. Q: Is person2 walking on a sidewalk and male?
A: no

15. Q: Is person2 walking on a sidewalk?
A: no

16. Q: Is person2 interacting with any other object?
A: yes

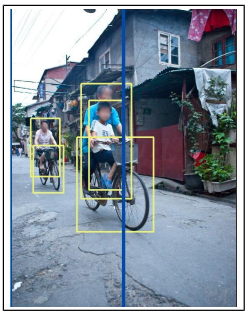
17. Q: Is there an interaction between person1 and person2?
A: no

18. Q: Is person1 taller than person2?
A: ambiguous

19. Q: Is person1 closer (to the camera) than person2?
A: no

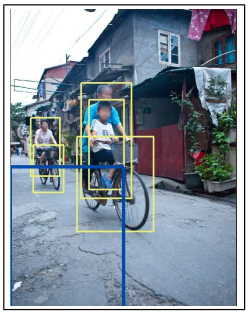
---

20. Q: Is there a person in the designated region?
A: yes



---

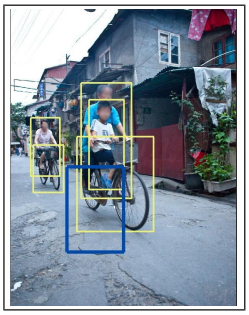
21. Q: Is there a unique person that is adult in the designated region?
A: no



---

22. Q: Is there a person that is male in the designated region?
A: yes

23. Q: Is there a unique person that is not carrying something in the designated region?
A: yes (person3)



---

24. Q: Is person3 walking on a sidewalk or sitting?
A: yes

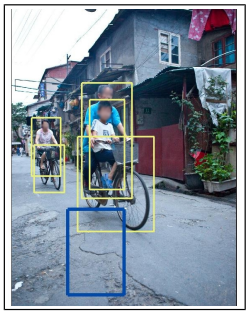
25. Q: Is person3 female and not wearing hat?
A: no

26. Q: Is person3 an adult and male?
A: no

27. Q: Is person3 interacting with any other object?
A: yes

---

28. Q: Are both person3 and person1 partially visible in the designated region?
A: no



---

29. Q: Is person1 taller than person3?
A: yes

30. Q: Is person1 closer (to the camera) than person3?
A: no

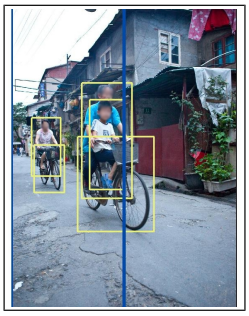
31. Q: Is there an interaction between person2 and person3?
A: no

32. Q: Is person2 taller than person3?
A: yes

33. Q: Is person2 closer (to the camera) than person3?
A: no

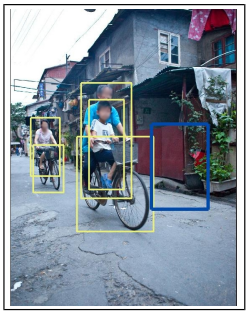
---

34. Q: Is there a person in the designated region?
A: no



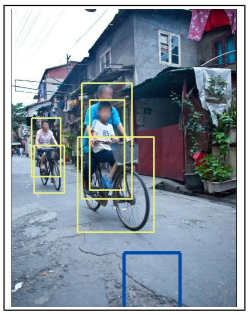
---

35. Q: Is there a vehicle in the designated region?
A: no



---

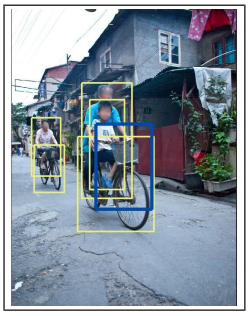
36. Q: Is there a vehicle in the designated region?
A: no



---

37. Q: Is there a vehicle in the designated region?
A: yes

38. Q: Is there a unique vehicle in the designated region?
A: yes (vehicle1)



---

39. Q: Does vehicle1 have no visible tires or have three or more visible tires?
A: no

40. Q: Is vehicle1 dark-colored?
A: yes

41. Q: Is vehicle1 parked?
A: no

42. Q: Does vehicle1 have only one visible tire?
A: no

43. Q: Is vehicle1 moving and a car?
A: no

44. Q: Is vehicle1 a car?

A: no

45. Q: Is vehicle1 moving?
A: yes

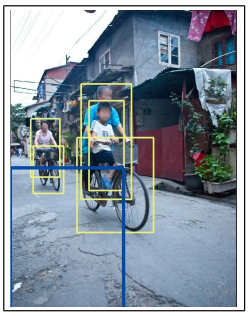
46. Q: Is vehicle1 a truck or a bicycle?
A: yes

47. Q: Is vehicle1 a truck?
A: no

48. Q: Is vehicle1 interacting with any other object?
A: yes

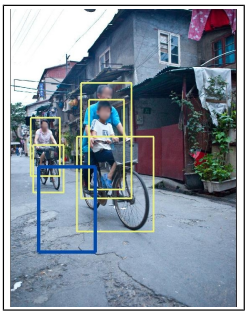
---

49. Q: Are both vehicle1 and person1 partially visible in the designated region?
A: yes



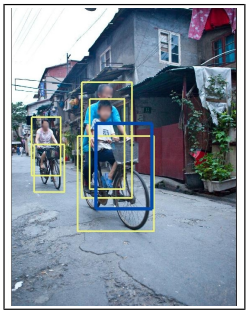
---

50. Q: Are both vehicle1 and person2 partially visible in the designated region?
A: yes



---

51. Q: Are both vehicle1 and person3 partially visible in the designated region?
A: yes



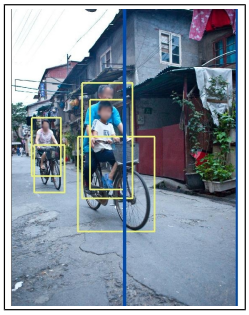
---

52. Q: Is there an interaction between vehicle1 and person3?
A: yes

53. Q: Is vehicle1 occluding person3?
A: yes

---

54. Q: Is there a person in the designated region?
A: no

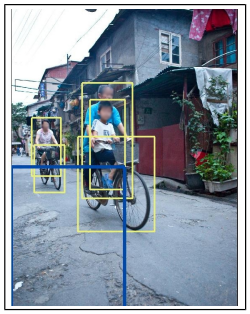


---

55. Q: Is there a vehicle that is dark-colored in the designated region?
A: yes

56. Q: Is there a unique vehicle that is a car, dark-colored in the designated region?
A: no

57. Q: Is there a unique vehicle that has two visible tires, is dark-colored in the designated region?
A: yes (vehicle2)



---

58. Q: Is vehicle2 parked and a car?
A: no

59. Q: Is vehicle2 moving?
A: yes

60. Q: Is vehicle2 interacting with any other object?
A: yes

61. Q: Is there an interaction between vehicle1 and vehicle2?
A: yes

62. Q: Is there an occlusion between vehicle1 and vehicle2?
A: no

63. Q: Is vehicle1 immediately behind vehicle2?
A: no

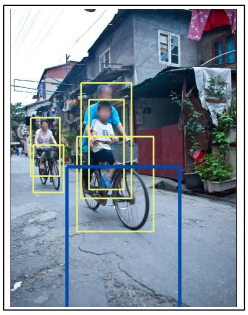
64. Q: Is vehicle1 larger than vehicle2?
A: ambiguous

65. Q: Is vehicle1 closer (to the camera) than vehicle2?
A: yes

---

66. Q: Are both vehicle2 and person1 partially visible in the designated region?

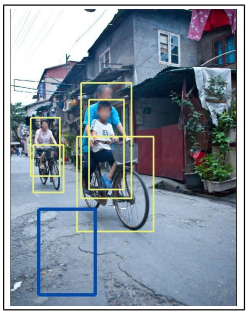
A: no



---

67. Q: Are both vehicle2 and person2 partially visible in the designated region?

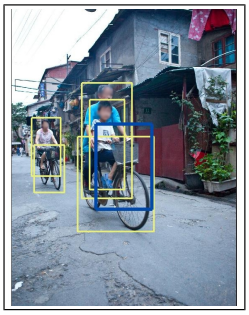
A: no



---

68. Q: Are both vehicle2 and person3 partially visible in the designated region?

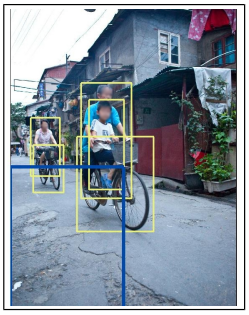
A: no



---

69. Q: Is there a vehicle that is car in the designated region?

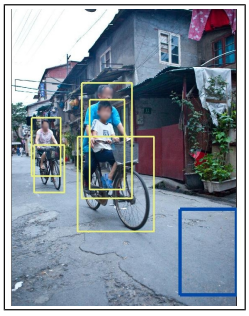
A: no



---

70. Q: Is there a vehicle in the designated region?

A: no



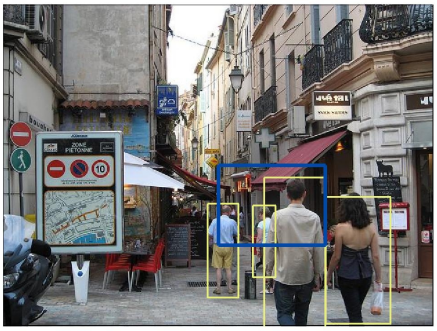
A.1.3 Example 3

---

1. **Q:** Is there a person in the designated region?
   A: yes

2. **Q:** Is there a unique person in the designated region?
   A: no

3. **Q:** Is there a unique person that is female in the designated region?
   A: yes (person1)



---

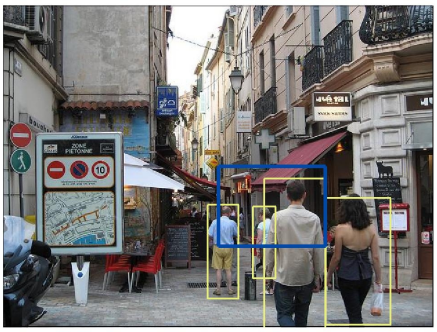
4. **Q:** Is person1 carrying something?
   A: ambiguous

5. **Q:** Is person1 walking on a sidewalk?
   A: no

6. **Q:** Is person1 crossing a street?
   A: no

7. **Q:** Is person1 an adult and standing still?
   A: yes

8. **Q:** Is person1 interacting with any other object?
   A: yes



9. **Q:** Is there a person in the designated region?
   A: yes

10. **Q:** Is there a unique person in the designated region?
    A: no

11. **Q:** Is there a person that is female in the designated region?
    A: no

12. **Q:** Is there a person that is walking on a sidewalk in the designated region?
    A: yes

13. **Q:** Is there a unique person that is walking on a sidewalk in the designated region?
    A: yes (person2)

---

14. **Q:** Is person2 female?
    A: no

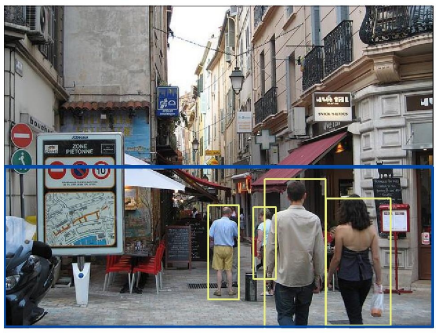
15. **Q:** Is person2 carrying something?
    A: no

16. **Q:** Is person2 interacting with any other object?
    A: yes

17. **Q:** Is there an interaction between person1 and person2?
    A: no

18. **Q:** Is person1 taller than person2?
    A: no

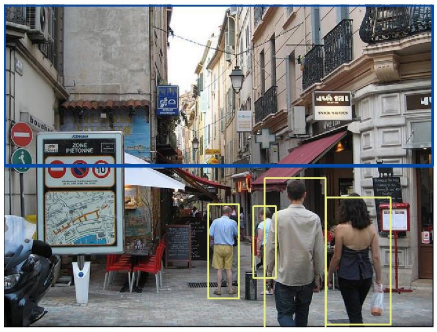
19. **Q:** Is person1 closer (to the camera) than person2?
    A: no



20. **Q:** Is there a vehicle with only one visible tire in the designated region?
    A: no

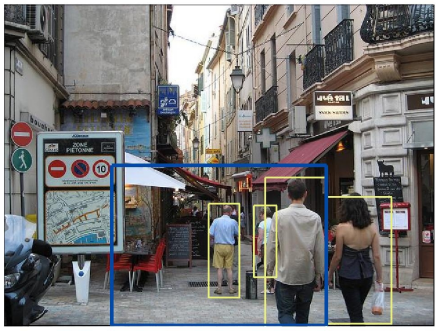
---

21. **Q:** Is there a vehicle in the designated region?
A: no



---

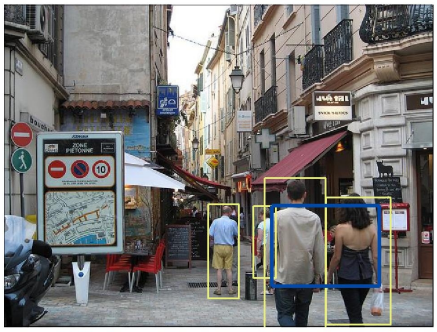
22. **Q:** Is there a vehicle in the designated region?
A: no



---

23. **Q:** Is there a person in the designated region?
A: yes

24. **Q:** Is there a unique person in the designated region?
A: yes (person3)



---

25. **Q:** Is person3 carrying something?
A: yes

26. **Q:** Is person3 walking on a sidewalk?
A: yes

27. **Q:** Is person3 male?
A: no

28. **Q:** Is person3 interacting with any other object?
A: yes

29. **Q:** Is there an interaction between person1 and person3?
    A: no

30. **Q:** Is person1 taller than person3?
    A: no

31. **Q:** Is person1 closer (to the camera) than person3?
    A: no

32. **Q:** Is there an interaction between person2 and person3?
    A: yes

33. **Q:** Are person2 and person3 talking?
    A: yes

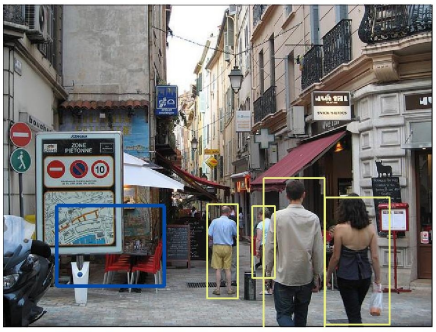
34. **Q:** Are person2 and person3 walking together?
    A: yes

35. **Q:** Is person2 taller than person3?
    A: yes

36. **Q:** Is person2 closer (to the camera) than person3?
    A: yes

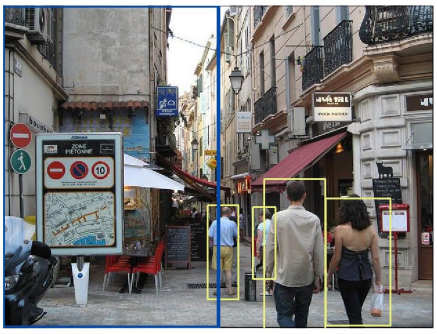
---

37. **Q:** Is there a person in the designated region?
    A: no



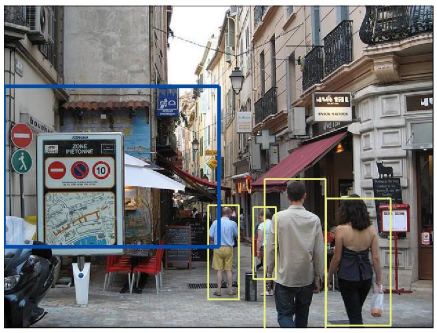
---

38. **Q:** Is there a person in the designated region?
    A: yes



---

39. **Q:** Is there a unique person in the designated region?
    A: yes (person4)



---

40. **Q:** Is person4 male?
    A: yes

41. **Q:** Is person4 walking on a sidewalk?
    A: no

42. **Q:** Is person4 carrying something?
    A: no

43. **Q:** Is person4 standing still?
    A: yes

44. **Q:** Is person4 interacting with any other object?
    A: yes

45. **Q:** Is person1 taller than person4?
    A: no

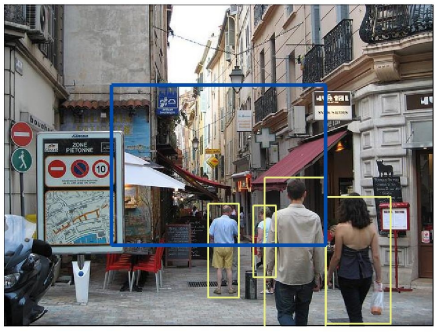
46. **Q:** Is person1 closer (to the camera) than person4?
    A: yes

47. **Q:** Is person2 taller than person4?
    A: yes

48. **Q:** Is person2 closer (to the camera) than person4?
    A: yes

---

49. **Q:** Are both person4 and person3 partially visible in the designated region?
    A: no



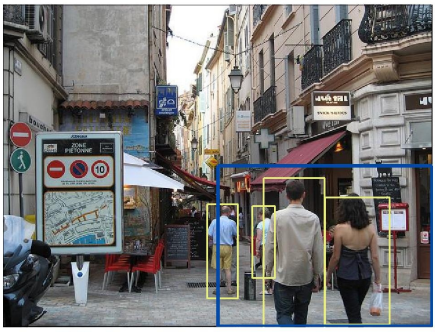
---

50. **Q:** Is person3 taller than person4?
A: no

51. **Q:** Is person3 closer (to the camera) than person4?
A: yes

---

52. **Q:** Is there a person in the designated region?
A: no



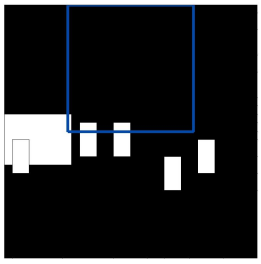
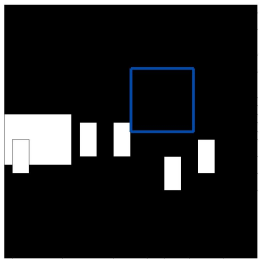
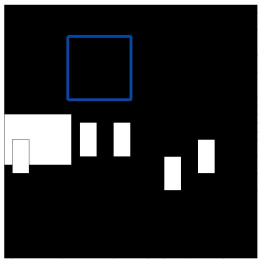
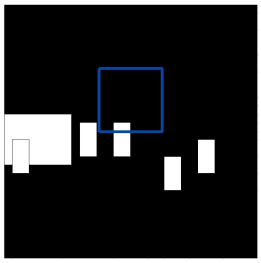
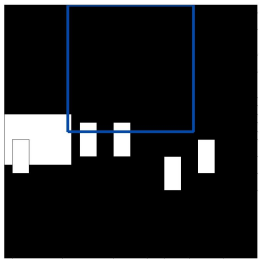
A.2 Rigorous Estimation

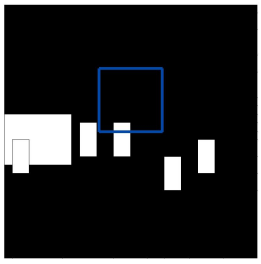
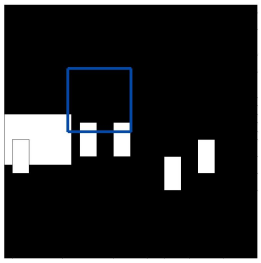
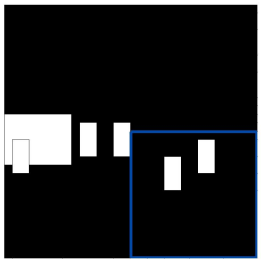
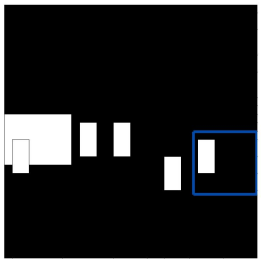
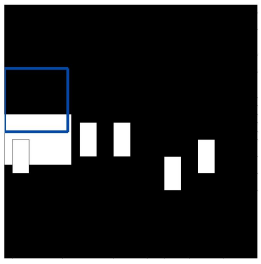
In this section, we display question sequences produced from our implementation of the visual Turing test in chapter 6. For readability, we display the questions in a colloquial form, but localization questions are understood to include the qualifier "partially visible" and instantiation questions are understood to include the qualifier "not previously instantiated." To the right of the localization questions we have inserted a small copy of the image, delineating the designated region with thick blue lines.

A.2.1 Example 1

In our first example, we display a question sequence corresponding to the rectangle discovery problem described in section 6.2. For a particular rectangle set R, we display a sequence of unpredictable questions and their answers. In these questions, people are associated with vertical rectangles, and vehicles associated with horizontal ones.

APPENDIX A. QUESTION STREAMS

Q1: Is there a person in the designated region?; A:1	
Q2: Is there a unique person in the designated region?; A:0	
Q3: Is there a unique person in the designated region?; A:0	
Q4: Is there a unique person in the designated region?; A:1	
Q5: Is there a vehicle in the designated region?; A:1	

Q6: Is there a unique vehicle in the designated region?; A:0	
Q7: Is there a unique vehicle in the designated region?; A:1	
Q8: Is there a vehicle in the designated region?; A:0	
Q9: Is there a unique person in the designated region?; A:1	
Q10: Is there a unique person in the designated region?; A:0	

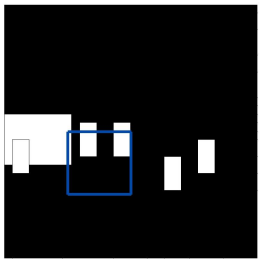
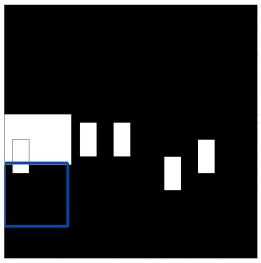
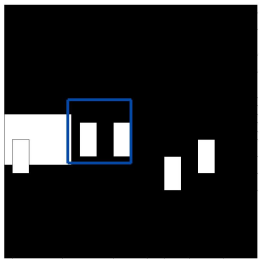
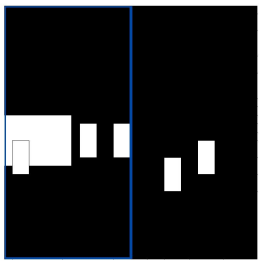
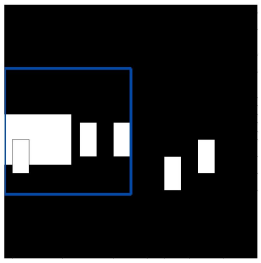
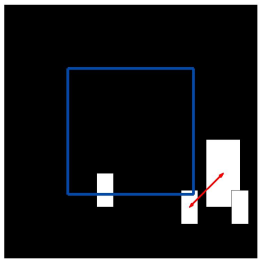
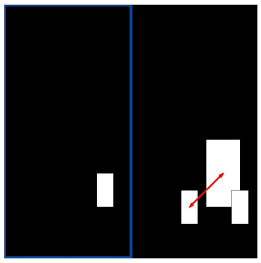
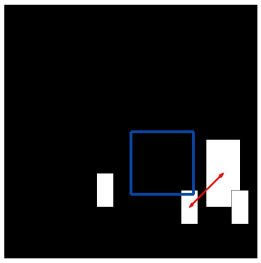
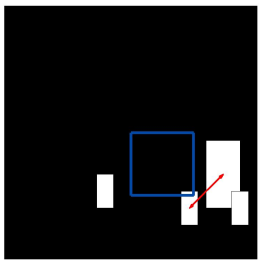
Q11: Is there a person in the designated region?; A:1	
Q12: Is there a person in the designated region?; A:1	
Q13: Is there a unique person in the designated region?; A:1	
Q14: Is there a unique person in the designated region?; A:1	
Q15: Is there a vehicle in the designated region?; A:0	

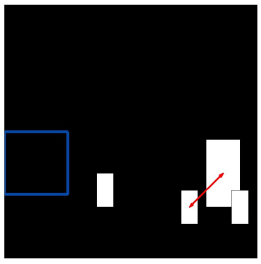
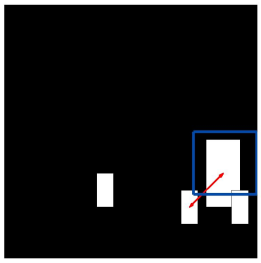
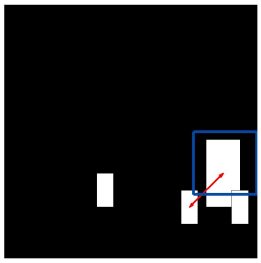
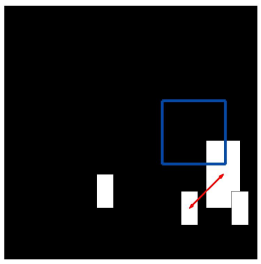
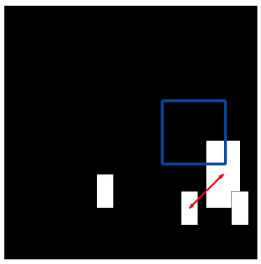
Table A.1: In the left column is a history $H_k = (F_k, A_k)$ of questions and answers. In the right column is an example scene cohering with the history.

A.2.2 Example 2

In our second example, we display a question sequence corresponding to the graph discovery problem described in section 6.3. For a particular graph G, we display a sequence of unpredictable questions and their answers.

APPENDIX A. QUESTION STREAMS

Q1: Is there a vehicle in the designated region?; A:0	
Q2: Is there a vehicle in the designated region?; A:0	
Q3: Is there a person in the designated region?; A:1	
Q4: Is there a unique person in the designated region?; A:1 (person1)	
Q5: Is person1 interacting with any other object?; A:1	

Q6: Is there a person in the designated region?; A:0	
Q7: Is there a person in the designated region?; A:1	
Q8: Is there a unique person in the designated region?; A:0	
Q9: Is there a person in the designated region?; A:1	
Q10: Is there a unique person in the designated region?; A:1 (person2)	

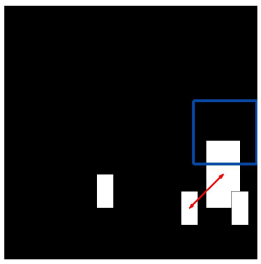
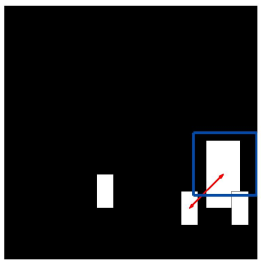
Q11: Is person2 interacting with any other object?; A:1	
Q12: Is person2 interacting with person1?; A:1	
Q13: Is there a unique person in the designated region?; A:0	
Q14: Is there a unique person in the designated region?; A:1 (person3)	
Q15: Is person3 interacting with any other object?; A:0	

Table A.2: In the left column is a history $H_k = (F_k, A_k)$ of questions and answers. In the right column is an example scene cohering with the history.

Bibliography

[1] Airoldi, Edoardo M., David M. Blei, Stephen E. Fienberg, and Eric P. Xing. ”Mixed membership stochastic blockmodels.” In Advances in Neural Information Processing Systems, pp. 33-40. 2009.

[2] Amit Y, Trouvé A (2007) Pop: Patchwork of parts models for object recognition. Int J Comput Vis 75:267-282.

[3] Baddeley, A. J. Likelihoods and pseudolikelihoods for Markov spatial processes. State of the art in probability and statistics, 21–49, Institute of Mathematical Statistics, Beachwood, OH, 2001.

[4] Bhamidi, Shankar, Guy Bresler, and Allan Sly. ”Mixing time of exponential random graphs.” Foundations of Computer Science, 2008. FOCS’08. IEEE 49th Annual IEEE Symposium on. IEEE, 2008.

[5] Chang LB, Jin Y, Zhang W, Borenstein E, Geman S (2011) Context, computation, and optimal roc performance in hierarchical models. Int J Comput Vis 93(2):117-140.

BIBLIOGRAPHY

[6] Chatterjee, Sourav, and Persi Diaconis. ”Estimating and understanding exponential random graph models.” The Annals of Statistics 41.5 (2013): 2428-2461.

[7] Cover, Thomas M., and Joy A. Thomas. Elements of information theory. John Wiley & Sons, 2012.

[8] Deng J, et al. (2009) Imagenet: A large-scale hierarchical image database. Proceedings IEEE 2009 CVPR (IEEE, New York), pp 248-255.

[9] Dowe, David L., and A. R. Hajek. ”A computational extension to the Turing Test.” Proceedings of the 4th Conference of the Australasian Cognitive Science Society, University of Newcastle, NSW, Australia. Vol. 1. No. 3. 1997.

[10] Endres I, Farhadi A, Hoiem D, Forsyth DA (2010) The benefits and challenges of collecting richer object annotations. Proceedings IEEE 2010 CVPR (IEEE, New York), pp 1-8. 27.

[11] Erdős, Paul & Rényi, Alfred. “On Random Graphs I”. Publ. Math. Debrecen, 6:290-297, 1959.

[12] Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. Int J Comput Vis 88:303-338.

[13] Fei-Fei L, Fergus R, Perona P (2003) A Bayesian approach to unsupervised one-shot learning of object categories. Proceedings IEEE 2003 ICCV (IEEE, New York), pp 1134-1141.

BIBLIOGRAPHY

[14] Felzenszwalb PF, Huttenlocher DP (2005) Pictorial structures for object recognition. Int J Comput Vis 61(1):55-79.

[15] Felzenszwalb PF, Girshick RB, McAllester D, Ramanan D (2010) Object detection with discriminatively trained part-based models. IEEE Trans Patt Anal Machine Intell 32: 1627-1645.

[16] Friedman N, Getoor L, Koller D, Pfeffer A. (1999) "Learning probabilistic relational models". In: International joint conferences on artificial intelligence, 1300?09

[17] Geman, D., Geman, S., Hallonquist, N., & Younes, L. (2015). Visual Turing test for computer vision systems. Proceedings of the National Academy of Sciences, 112(12), 3618-3623.

[18] Getoor, Lise and Taskar, Ben. Introduction to statistical relational learning, MIT Press, 2007

[19] E.N. Gilbert. Random Graphs. Ann. Math. Statist., 30:1141-1144, 1959.

[20] Girshick R, Donahue J, Darrell T, Malik J (2013) Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524.

[21] Grenander, Ulf, Lectures in Pattern Theory I, II and III: Pattern Analysis, Pattern Synthesis and Regular Structures, Springer-Verlag, Heidelberg-New York, 1976-1981.

[22] Grenander, Ulf. Elements of pattern theory. JHU Press, 1996.

[23] Grenander, Ulf. ”Geometries of knowledge.” Proceedings of the National Academy of Sciences 94.3 (1997): 783-789.

[24] Grenander, Ulf, and Michael I. Miller. Pattern theory: from representation to inference. Vol. 1. Oxford: Oxford university press, 2007.

[25] Griffin, Gregory and Holub, Alex and Perona, Pietro (2007) Caltech-256 Object Category Dataset. California Institute of Technology

[26] Handcock, Mark S., et al. Assessing degeneracy in statistical models of social networks. Vol. 39. Working paper, 2003.

[27] Hariharan B, Arbelez P, Girshick R, Malik J (2014) Simultaneous Detection and Segmentation. ECCV 2014 (Springer, Heidelberg, Germany), pp 297-312.

[28] Harnad, Stevan. ”Other bodies, other minds: A machine incarnation of an old philosophical problem.” Minds and Machines 1.1 (1991): 43-54.

[29] D. Heckerman, C. Meek, and D. Koller. Probabilistic Entity-Relationship Models, PRMs, and Plate Models. In Getoor and Taskar, 2007

[30] Hernandez-Orallo, Jose, and David L. Dowe. ”Measuring universal intelligence: Towards an anytime intelligence test.” Artificial Intelligence 174.18 (2010): 1508-1539.

[31] Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527-1554.

[32] Hoff, Peter D., Adrian E. Raftery, and Mark S. Handcock. ”Latent space approaches to social network analysis.” Journal of the american Statistical association 97.460 (2002): 1090-1098.

[33] Holland, Paul W., and Samuel Leinhardt. ”An exponential family of probability distributions for directed graphs.” Journal of the american Statistical association 76.373 (1981): 33-50.

[34] Holland, Paul W., Kathryn Blackmond Laskey, and Samuel Leinhardt. ”Stochastic blockmodels: First steps.” Social networks 5.2 (1983): 109-137.

[35] Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. NIPS (Neural Information Processing Systems Foundation, La Jolla, CA), pp 1097-1105.

[36] Latouche, Pierre, Etienne Birmel, and Christophe Ambroise. ”Overlapping stochastic block models with application to the french political blogosphere.” The Annals of Applied Statistics (2011): 309-336.

[37] Legg, Shane & Hutter, Marcus (2007). Universal intelligence: A definition of machine intelligence. Minds and Machines 17 (4):391-444.

[38] Lowe DG (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2):91-110.

BIBLIOGRAPHY

[39] Lu W, Lian X, Yuille A (2014) Parsing semantic parts of cars using graphical models and segment appearance consistency. arXiv:1406.2375.

[40] Nowicki, Krzysztof, and Tom A. B. Snijders. "Estimation and prediction for stochastic blockstructures." Journal of the American Statistical Association 96.455 (2001): 1077-1087.

[41] Oh S, et al. (2011) A large-scale benchmark dataset for event recognition in surveillance video. Proceedings IEEE 2011 CVPR (IEEE, New York), pp 3153-3160.

[42] Ommer B, Sauter M, Buhmann JM (2006) Learning top-down grouping of compositional hierarchies for recognition. Proceedings IEEE 2006 CVPR (IEEE, New York), pp 194-201.

[43] Open Connectome Project. (2014, November 15). Retrieved from http://openconnecto.me/graph-services/download/

[44] Oquab M, Bottou L, Laptev I, Sivic J, et al. (2014) Learning and transferring mid-level image representations using convolutional neural networks. Proceedings IEEE 2014 CVPR (IEEE, New York), pp 1717-1724.

[45] Ove, Frank and Strauss, David. "Markov graphs." Journal of the american Statistical association 81.395 (1986): 832-842.

[46] Ozdemir B, Aksoy S, Eckert S, Pesaresi M, Ehrlich D (2010) Performance measures for object detection evaluation. Pattern Recognit Lett 31:1128-1137.

BIBLIOGRAPHY

[47] Ripley, Brian D., and Frank P. Kelly. ”Markov point processes.” Journal of the London Mathematical Society 2.1 (1977): 188-192.

[48] Robins, Garry. ”Exponential random graph models for social networks.” Handbook of Social Network Analysis. Sage (2011).

[49] Russell BC, Torralba A, Murphy KP, Freeman WT (2008) Labelme: A database and web- based tool for image annotation. Int J Comput Vis 77(1-3):157-173.

[50] Russell SJ, Norvig P (2003) Artificial Intelligence: A Modern Approach (Pearson Education, Harlow, UK).

[51] Salakhutdinov, Ruslan R. ”Learning in Markov random fields using tempered transitions.” Advances in neural information processing systems. 2009.

[52] Saygin AP, Cicekli I, Akman V (2003) Turing Test: 50 Years Later. The Turing Test, ed Moor JH (Springer, Heidelberg, Germany), pp 23?78.

[53] Schweinberger, Michael. ”Instability, sensitivity, and degeneracy of discrete exponential families.” Journal of the American Statistical Association 106.496 (2011): 1361-1370.

[54] Schweinberger, Michael, and Mark S. Handcock. ”Local dependence in random graph models: characterization, properties and statistical inference.” Journal of the Royal Statistical Society: Series B (Statistical Methodology) 77.3 (2015): 647-676.

[55] Shalizi, Cosma Rohilla, and Alessandro Rinaldo. ”Consistency under sampling of exponential random graph models.” The Annals of Statistics 41.2 (2013): 508-535.

[56] Shervashidze; Schweitzer; Jan van Leeuwen; Mehlhorn; Borgwardt; “Weisfeiler-Lehman Graph Kernels”; Journal of Machine Learning Research 12 (2011) 2539-2561

[57] Searle, John R. ”Minds, brains, and programs.” Behavioral and brain sciences 3.03 (1980): 417-424.

[58] Snijders, Tom AB, et al. ”New specifications for exponential random graph models.” Sociological methodology 36.1 (2006): 99-153.

[59] Strauss, David. ”On a general class of models for interaction.” SIAM review 28.4 (1986): 513-527.

[60] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2002.

[61] Turing AM (1950) Computing machinery and intelligence. Mind 59:433?460.

[62] Wolpert, D. (1996). The lack of a prior distinctions between learning algorithms and the existence of a priori distinctions between learning algorithms. Neural Computation, 8:1341 1390,1391 1421

[63] Wolpert, David H., and William G. Macready. No free lunch theorems for search. Vol. 10. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

BIBLIOGRAPHY

[64] Wolpert, David H., and William G. Macready. ”No free lunch theorems for optimization.” Evolutionary Computation, IEEE Transactions on 1.1 (1997): 67-82.

[65] Xiao J, Hays J, Ehinger KA, Oliva A, Torralba A (2010) Sun database: Largescale scene recognition from abbey to zoo. Proceedings IEEE 2010 CVPR (IEEE, New York), pp 3485-3492.

[66] Yao B, Yang X, Zhu SC (2007) Introduction to a Large-Scale General Purpose Ground Truth Database: Methodology, Annotation Tool and Benchmarks (Springer), pp 169-183. 26.

[67] Younes, Laurent: Estimation and annealing for Gibbsian fields. Annales de l'Institut Henri Poincar, vol 24 No 2 269-294 (1988)

[68] Yu G,Morel JM (2009)A fully affine invariant image comparison method.Proceedings IEEE 2009 International Conference on Acoustics, Speech and Signal Processing (IEEE, New York), pp 1597-1600.

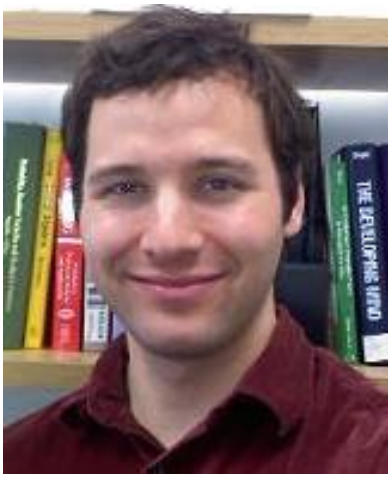
[69] Zhang N, Paluri M, Ranzato M, Darrell T, Bourdev L (2013) Panda: Pose aligned networks for deep attribute modeling. arXiv:1311.5591. 22.

[70] Zhu Q, Yeh MC, Cheng KT, Avidan S (2006) Fast human detection using a cascade of histograms of oriented gradients. Proceedings IEEE 2006 CVPR (IEEE, New York), Vol 2, pp 1491-1498.

BIBLIOGRAPHY

[71] Zhu SC, Mumford D (2006) A stochastic grammar of images. Foundations and Trends in Computer Graphics and Vision 2(4):259-362.

Vita



Neil F. Hallonquist received his bachelors degree in Electrical Engineering from Rensselaer Polytechnic Institute in 2004, a masters degree in Electrical Engineering from Johns Hopkins University in 2006, and enrolled in the Applied Mathematics and Statistics Ph.D. program at Johns Hopkins University in 2009, working at the Center for Imaging Science.